

Synthesis of Spatially Extended Sources in Virtual Reality Audio

Micah Hakala

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 8.7.2019

Supervisor

Prof. Ville Pulkki

Advisor

MSc. Juhani Paasonen

Author Micah Hakala

Title Synthesis of Spatially Extended Sources in Virtual Reality Audio

Degree programme Communication and Information Sciences

Major Acoustics and Audio Technology

Code of major ELEC3030

Supervisor Prof. Ville Pulkki

Advisor MSc. Juhani Paasonen

Date 8.7.2019

Number of pages 68+1

Language English

Abstract

This thesis details a real-time implementation of spatial extent synthesis for virtual sound source objects made from mono sound signals and source object geometries. Techniques for distributing components of sound across basic and mesh-like geometry surfaces are discussed. A virtual-world audio environment supporting a listener avatar and various spatially extensive sound sources is described, and forms of source-to-listener distance attenuation are outlined with their roles in sound localization of spatially extensive sound sources. The implementation described herein takes form as an audio plug-in, of which the behavior, usage details, and compatible host applications are mentioned.

Keywords spatial audio, spatial extent, ambisonics, virtual reality, time-frequency domain processing, 3D mesh, quasi-random sequences

Contents

Abstract	2
Contents	3
Acknowledgements	6
Abbreviations	7
1 Introduction	8
1.1 Motivation	9
1.2 This Thesis	11
2 Human Spatial Sound Perception	13
2.1 Introduction	13
2.2 Interaural Time Difference (ITD)	13
2.3 Interaural Level Difference (ILD)	15
2.4 Monaural Cues	16
2.5 Apparent Source Size	17
3 Time-Frequency Domain Audio Processing	19
3.1 Short-Time Fourier Transform in Real-Time	19
3.2 Overlap-Add Short-Time Fourier Transform	20
3.2.1 Alias-Free Short-Time Fourier Transform	23
4 Spatial Audio	24
4.1 Ambisonics and B-Format	24
4.1.1 Ambisonic Panner	25
4.2 Directional Audio Coding (DirAC)	26
4.3 Binaural HRTF Audio	27
4.4 Target Audio Encoding for This Thesis	28
5 Method for Spatial Extent Synthesis	29
5.1 Spatial Extent Synthesis Steps	30
5.1.1 Geometry Initialization	30

5.1.2	Scale, Rotation, Translation	31
5.1.3	Frequency Decomposition, Component Spatialization, and Mixing	31
5.1.4	Inverse Fourier Transform for Final-Mix B-Format Signal . . .	32
5.1.5	Soundfield Reproduction from Final-Mix B-Format Signal . .	32
5.2	Alternative Methods for Signal Decomposition	32
6	Point Distribution onto 3D Geometry	34
6.1	Halton Sequences	34
6.2	Primitive Shapes	35
6.3	Area-Based Application of Halton Sequences for 3D Mesh Sampling .	37
6.4	Strengths and Weaknesses of Surface Distribution	42
6.5	Opaque Sound Sources with Back-Face and Occlusion Culling	43
7	Distance Attenuation for Spatially Extended Sources in Virtual Environments	46
7.1	Hybrid Distance Attenuation Method	50
8	Implementation: Real-Time Virtual-World Spatial Extent Synthesizer	53
8.1	Real-Time Plug-In Architecture	53
8.2	VST and JUCE	54
8.3	Hosts	54
8.3.1	Reaper	54
8.3.2	Max	55
8.3.3	Unity (Game Engine)	56
8.4	Command-Interface and Virtual State	58
8.5	Global State and Parameters	59
8.6	Virtual Sound Source State and Parameters	60
8.6.1	State	60
8.6.2	Parameters	61
9	Conclusion	65
	Bibliography	66

A	Formulas	69
A.1	Rodrigues' rotation formula	69
A.2	Heron's formula	69

Acknowledgements

Thanks to:

Ville Pulkki, for the research opportunity that was offered leading to this thesis, for patiently awaiting this result, and for the instruction.

Juhani Paasonen, for all the direction, reading, and suggestions you made which have greatly improved this thesis.

Andrea Mancianti, for being my implementation's QA department.

Veli Laamanen, for driving the request for a Unity plug-in. The plug-in ended up helping immensely.

Mike Skrzypczak, for the permission to pursue this degree without affecting my work status and for getting me on the team in 2016.

Matt Durgavich, for all the time that you found amongst your crazy schedule to deal with my matters.

Aalto University's Acoustics and Audio/Speech Signal Processing professors

The Aalto Acoustics Lab researchers

Friends: J. Feely, N. Juenke, R. Hansen, L. Saikkonen, R. Falcón Pérez,
V. H. Souza

Lea Kosonen

Family

Duluth, MN 8.7.2019

Micah Hakala

Abbreviations

DAW	Digital Audio Workstation
DCT	Discrete Cosine Transform
DirAC	Directional Audio Coding
DoA	Direction of Arrival
DSP	Digital Signal Processing
DST	Discrete Sine Transform
FIR	Finite Impulse Response
HRIR	Head-Related Impulse Response
HRTF	Head-Related Transfer Function
IDE	Integrated Development Environment
ILD	Interaural Level Difference
ITD	Interaural Time Difference
MIDI	Musical Instrument Digital Interface
OLA	Overlap-Add
OMDirAC	Optimal Mixing Directional Audio Coding
OS	Operating System
SDK	Software Development Kit
SHlib	Spherical Harmonics library
TF-domain	Time-Frequency domain
UDP	User Datagram Protocol
UI	User Interface
VBAP	Vector Base Amplitude Panning
VST	Virtual Studio Technology
VW	Virtual World

1 Introduction

Since the dawn of real-time capable audio/visual computing, the realistic synthesis of sound scenes has been in ever increasing demand for the purposes of immersive virtual reality. Generally, the heights of virtual reality are considered a composite of all of its employable technologies: real-time computer graphics, sound reproduction, motion tracking, haptic feedback, and even odor dispensers. However, each of these technologies can be said to constitute some virtual reality in their own domains.

In the domain of acoustics, for example, a sound field that would be present at a concert-goer's head can be simulated for a listener at home on their couch by their home theater surround sound system. In this example not only would musical content be heard and the sonic character of an orchestra hall be gleaned, but the methods available provide sufficient details so that audiospatial sensing is induced in the listener. The location of certain instruments onstage can be deduced, reflections can be felt as arriving from the sides of the hall, and sounds that come from every which way surround the listener. Certainly this demonstration would create feelings for the listener of being in another time and place.

Various surround sound formats can accomplish this example's encompassing effect by encoding spatial sound fields for use with specific sound reproduction systems. Surround sound formats and loudspeaker setups such as 5.1 ("five-point one") and 7.1 ("seven-point one") are commonly available in digital home video releases and video games. However, a certain surround sound format promises the holy grail of spatial audio formats: potentially universal playback capability because of its independence from sound reproduction systems and its full-sphere surround sound encoding ability. That surround sound format is called 'ambisonics' (Gerzon, 1980). The sound field data required for the home theater orchestra scenario can be encoded into ambisonics directly from recordings that are taken from where a listener's head would be during a concert. The representation of the sound field in this directly recorded ambisonics format is known as 'B-format'. From this format, almost any sound reproduction system, from loudspeakers to headphones, can recreate acoustic scenes that carry perceivable spatial qualities in both horizontal and vertical dimensions.

Digital synthesis of these ambisonics B-format signals has been a popular topic of research for the purposes of rendering spatial audio in virtual reality applications. This is because we cannot simply record natural phenomena from a virtual environment; all sound must be generated somehow computationally. However, physical simulations of acoustic aspects in real-time rendered virtual environments are often not implemented because of their high computational cost (James et al., 2006). Certain processes of physically modeled acoustics can be precomputed (James et al., 2006) in order to amortize their computational costs, but there are still processes which cannot be precomputed. In those cases, convincing perceptual approximations, which are not necessarily physically accurate, are worthy of development. There is an analogue here between the study of computer graphics and this issue with computer simulated acoustics. In computer graphics, physically modeled ray-tracing has been difficult to perform adequately in real-time, but rasterization with so-called "shaders" has been

a feasible way of approximating visuals for graphical scenes that is both convincing to the eye and suitable for real-time use. Now, it isn't necessary to compute sound pressure levels for every point in space when creating convincing spatial sound approximations for virtual environments. Most of the time we only have to concern ourselves with what some listener should be able to hear. This brings to mind the famous thought experiment: "If a tree falls in a forest and no one is around to hear it, does it make a sound?"

There is one particular topic that this thesis goes into length about that arises from the development of spatial sound approximations for virtual worlds: the synthesis of spatial extent for non-trivially sized virtual sound source objects. It is known that humans can sense the spatial extent of auditory events to some degree (Blauert, 1997). This fact has led to experiments in the synthesis of spatial extent for virtual sound sources (Schmele and Sayin, 2018; Potard and Burnett, 2004; Pihlajamäki et al., 2014). In addition, sound source "size" (extent) has become a parameter available for configuration in some spatial audio software.

1.1 Motivation

Two of the original goals of this thesis work were:

1. To build a virtual-world spatial audio renderer using Aalto University's Acoustics Lab's existing software libraries.
2. To interface the virtual-world spatial audio renderer with an existing 3D graphics renderer.

These goals were accomplished and will be discussed in a section dedicated to the resulting implementation. However, the work ended up taking an even greater focus onto one particular aspect of the implemented audio renderer: its ability to spatially extend mono sound source signals using time-frequency decomposition. This ability for spatial extent synthesis is priceless for virtual-world sound designers because mono sound signals are abundant. Mono sound signals are easy to record, synthesize, and manipulate; unlike ambisonics signals. Indeed, this focus on spatial extent synthesis became the main subject for this thesis.

Of great relevance to the topic of this thesis is research that has gone into, none other than, the spatial extent synthesis of virtual sound sources using time-frequency decomposition of mono signals (Pihlajamäki et al., 2014). This paper by Pihlajamäki et al. laid the groundwork for this thesis. However, if an outsider who is not privy to the code base kept at Aalto University's Acoustics Lab were to research this topic, they would be unknowing of relevant features that have not been published. There are relevant capabilities of a virtual-world audio renderer, named VW-DirAC (Pihlajamäki et al., 2013), which the acoustics group at Aalto University has created.

VW-DirAC’s capabilities for mono signal spatial extent synthesis in a virtual world are that:

- Mono signals can be decomposed into time-frequency domain components.
- Frequency components of mono signals can be evenly distributed within the volumes of cuboids (cubes and axis-scaled cubes), spheres, and cylinders.
- Sound sources and a listener (or avatar) can move in real-time.
- When the listener is outside of an extended sound source, the source level will be rendered at lower volumes. This loudness reduction, known as ‘distance attenuation’, is calculated from the distance between the listener position and the closest point on the analytical surface of the cuboid, sphere, or cylinder geometry.
- When the listener is inside of an extended sound source, then there is no distance attenuation and the audio gain is set for full volume.
- Synthesized audio is at first encoded as ambisonics B-format signals, but ultimately VW-DirAC outputs loudspeaker signals.

The full list of features differs from what could be inferred from only the publication of Pihlajamäki et al. (2014). Instead of extending sounds onto a ring of loudspeakers or a sector of a ring like in their listening tests, three types of geometry are available that can provide vertical extent in addition to horizontal extent. In their listening tests, listeners were allowed to swivel, but neither the listeners nor the sound sources could change positions. However, it was already possible at the time for virtual listeners and sound sources to move in real-time.

After surveying the state of this topic, I found that there were some limitations of mono signal spatial extent synthesis in VW-DirAC and Pihlajamäki et al. (2014):

- In VW-DirAC, spatial extent distribution can be applied for only three simple 3D shapes (cuboid, sphere, and cylinder). There has been no proposal for generalizing spatial distribution for arbitrary shapes in this topic’s context.
- They had not considered acoustically opaque virtual sound source objects. In other words, solid acoustic objects where the closest hull shadows surfaces that are behind it. VW-DirAC spatially distributes time-frequency components volumetrically. Without shadowing, sound source surfaces are acoustically “transparent”. The transparency in this case would lead to spatial energy density inconsistencies.
- Distance attenuation conventions for concave geometries have not been researched.

I was attracted by these oversights and uncharted territories. After completing the basis of a virtual world spatial audio renderer that would meet the original goals of this thesis work, my research set out on a path to see what these issues with spatial extent synthesis would reveal.

1.2 This Thesis

This thesis presents an implementation of a real-time virtual-world audio renderer. This audio renderer is configurable and capable of maintaining listener and sound source orientations. It accepts mono sound signals as audio inputs and spatializes them as either point sources or spatially extensive sources, and the results are output as ambisonics B-format audio signals. The audio renderer is available for use in some common audio plug-in hosts and with two popular 3D renderers, a videogame engine called Unity and the Jitter video extensions for the multimedia graphical-programming runtime environment Max. It is implemented in the C++ programming language using a plug-in architecture to drive real-time synthesis. UDP sockets provide inter-process communication for scriptable writing and reading of the audio renderer’s virtual-world state and parameters.

Beyond the audio renderer basis, this thesis delves into the specifics of unexplored topics for mono signal spatial extent synthesis with time-frequency decomposition.

A goal of this thesis is to spatially extend sound sources across arbitrary 3D meshes in order to intrinsically synthesize approximations of sound fields that carry spatial extent information. Sound source extents will be made congruent to the area that virtual sound source objects of their certain shapes would precisely project to a listener rather than limiting spatial distribution to the three types of primitive geometries provided by VW-DirAC. This will be made possible whether a listener is inside or outside of the bounds of arbitrary sound source geometry.

This thesis discusses the synthesis of opaque virtual sound source objects where it is assumed that only the apparent face of a shape is important for spatial extent synthesis of anechoic sound.

This thesis mentions how different means for calculating the distance attenuation of spatialized sound sources are important to the perception of spatial extent in situations where listeners and sound sources can move in real-time. Also, complications with spatial extent synthesis for concave geometry will be discussed and an experimental method is proposed which attempts to mitigate the stated complications.

The scope of this thesis paper will not include evaluation of the acoustically reproduced sound fields generated from the audio engine in a loudspeaker environment or with headphones. The scope will also not include synthesis of spatial reverberation, although it was fielded during development because of its potential for creating more immersive virtual environments.

The remainder of this thesis paper is structured as follows.

Chapter 2 outlines relevant aspects of human spatial sound perception.

Chapter 3 introduces the basics of time-frequency domain audio processing and how it can be applied in real-time.

Chapter 4 summarizes the theory behind a certain surround sound spatial audio encoding and some of its decoders. Ambisonics B-format, Directional Audio Coding (DirAC), and binaural head-related transform audio are discussed.

Chapter 5 presents the main technique used for rendering spatial extent synthesis into ambisonics B-format signals.

Chapter 6 evaluates quasi-random distribution of points across primitive 3D geometry and proposes a method for quasi-random point distribution across arbitrary 3D meshes.

Chapter 7 considers some methods of distance attenuation for spatially extended sound sources in virtual worlds and highlights their strengths and weaknesses.

Chapter 8 goes into detail about the virtual-world audio engine; covering areas such as its plug-in architecture, utilization in VST hosts and the Unity game engine, its command interface, state, parameters, features, and behaviors.

Chapter 9 concludes the thesis by reflecting on what was achieved that met the goals and aims set for this research. Potential improvements to the implementation are discussed as well.

2 Human Spatial Sound Perception

This thesis sets out to synthesize spatial qualities of extended sound sources into an audio format which will preserve spatial data. It must be understood what spatial qualities humans can hear and what physical phenomena can induce their perception so that effective audio encoding and decoding are chosen.

2.1 Introduction

Humans and other animals are capable of perceiving spatial details from their sense of hearing. Localization of audio events is primarily discussed in literature, but other perceptions exist such as detection of occlusion (Farag et al., 2003), detection of sound source movements, and inference of room spaciousness from reverberation characteristics.

Spatial hearing occurs when a human receives certain cues to an audiospatial system in their brain. These cues can be caused by transduction from both ears, binaural cues, or transduction at one ear, monaural cues (Yost and Gourevitch, 1987). The primary binaural cues are interaural time difference (ITD) and interaural level difference (ILD); both of which occur when a sound source is closer to one ear than the other. The primary monaural cue is a coloration of the frequency spectrum for received sound due to acoustic wavefront interactions with a listener's body; namely the head, shoulders, and pinnae (Ahveninen et al., 2014).

The sum total effect of binaural and monaural cues which are received at a listener's ear canals can be described by transfer functions for each ear given a sound source location relative to the listener. These transfer functions are called 'Head-Related Transfer Functions' or HRTFs. More detail on HRTFs is written in Chapter 4.

2.2 Interaural Time Difference (ITD)

When a sound source is closer to one ear than the other, the closer ear will hear sound earlier than the farther ear. This happens because the acoustic wavefronts take time to propagate. When a sound occurs on a listener's median plane, then the acoustic wavefronts will arrive to both ears simultaneously, barring certain asymmetries of the listener's ears on their head.

Subtle time differences between the sounds arriving at each ear create a cue that is used to localize sound sources on the horizontal plane. Time differences can be as long as $630 \mu\text{s}$ for a head with ears distanced 21.5 cm apart and a speed of sound at 343 meters per second given typical atmospheric conditions. Humans can make use of interaural time differences that are as low as $10 \mu\text{s}$ (Wang and Brown, 2006). In a set of HRTFs viewed in the time-domain, differences in pressure onsets can be seen between the transfer functions for left and right ears taken from pairs of functions intended for sound source locations that are off of the listener's median plane.

An illustration of interaural time difference is displayed in Figure 2.1. The first frame shows the position where an acoustic impulse has occurred as a green dot. This impulse could have been caused by a small firecracker or bubble popping, for instance. The impulse occurred up and to the right of the listener's head, so we should expect that the listener's right ear will hear the impulse first. The second frame shows how the impulse expands in space before reaching the listener's body. The third frame shows that the right ear indeed hears the impulse first while there is no pressure deviations near the left ear yet. Finally, the fourth frame shows when the left ear hears the impulse after the right ear.

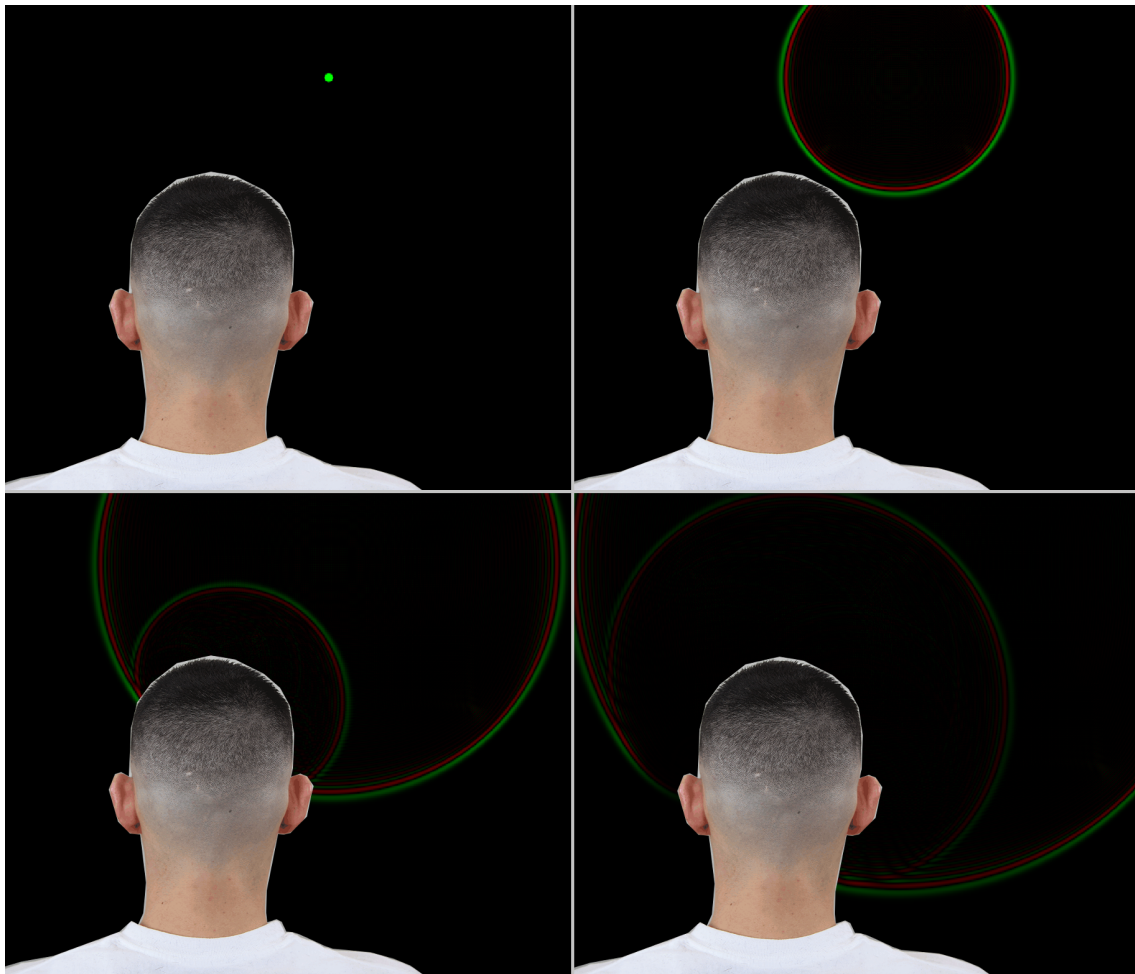


Figure 2.1: An impulse occurs up and to the right of a listener. Green is high pressure, black is ambient pressure, and red is low pressure. The first frame shows the starting position of the impulse. The second frame shows the expanding impulse before it has reached the listener's body. The third frame shows the moment when the impulse can first be heard by the listener through his right ear. The final frame shows the first moment when the listener's left ear can hear the impulse.

2.3 Interaural Level Difference (ILD)

Also when a sound source is closer to one ear than the other, the closer ear will hear sound as louder than the farther ear. This happens primarily because the acoustic wavefronts are shadowed by the listener's head, so that arriving waves are partially absorbed, reflected, and diffracted.

The difference in sound levels vary by frequency because absorption and diffraction are frequency dependent in acoustic sound propagation. Decently low frequencies will propagate around the head because the head's geometry is much smaller than the frequencies' quarter wavelengths, resulting in little to no level difference. For much higher frequencies the sound will not propagate directly to the other ear because it is mostly reflected off of the body. For frequencies in between those two extremes, sound will propagate to the other ear, but the general rule of thumb is: the higher the frequency, the quieter it will be to the shadowed ear.

Spectral sound level differences create another cue that is used to localize sound sources on the horizontal plane. It is ILD and ITD that were the basis for Lord Rayleigh's Duplex Theory of interaural difference based sound localization (Rayleigh, 1907). In a set of HRTFs, level differences can be seen between transfer functions for left and right ears taken from pairs of functions intended for sound source locations that are off of the listener's median plane.

An illustration of interaural level difference is displayed in Figure 2.2. Both frames show point sources that are emitting single frequencies. The left frame has a point source emitting 1.715 kHz and the right frame has a point source emitting 6.68 kHz. The left side of the listener's head sees lower sound pressures from both sound sources than the right side of the head sees. However, the lower frequency source produces higher sound pressure than the higher frequency source at the left side of the listener's head. This demonstrates the head shadowing effect that higher frequencies are more affected by.

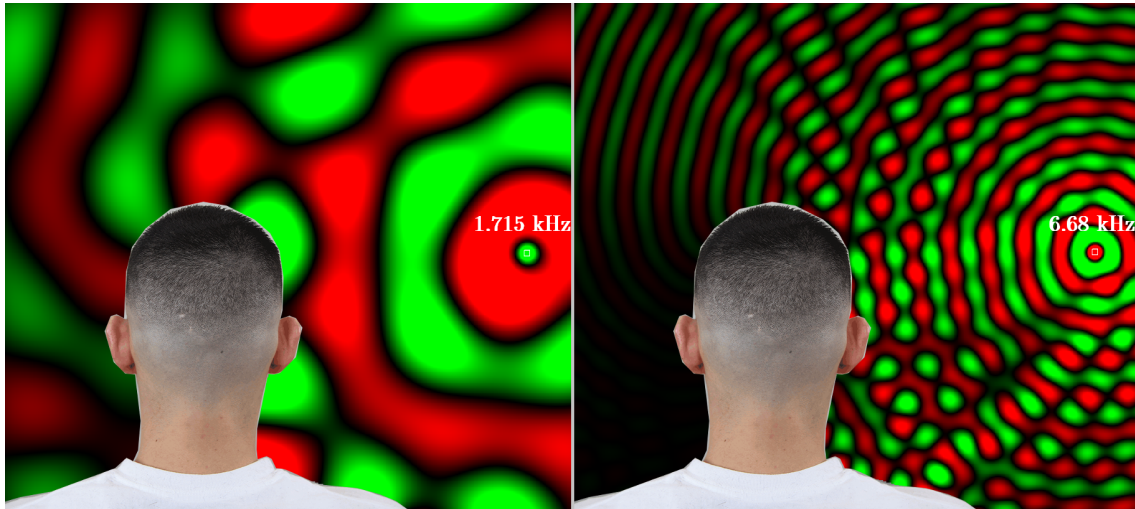


Figure 2.2: Two point sources emitting single frequencies (left: 1.715 kHz, right: 6.68 kHz) are shown interacting with a listener's body. Green is high pressure, black is ambient pressure, and red is low pressure. Both frequencies have reduced sound pressures on the left side of the head, but the lower frequency (left) has higher sound pressures on the left side of the head compared to what the higher frequency (right) produces.

2.4 Monaural Cues

When sound reflects off of surfaces, it can interact with later produced waveforms that are still arriving. As this occurs, sound pressures at spaces where sound is still arriving will fluctuate above and below what would have occurred without an interaction with a reflected sound. In other words, constructive and destructive self-interference occurs which creates resonances and nulls or higher and lower sound pressure level regions, respectively. The regions where either constructive or destructive interference occur are at different places for different frequencies because they are dictated by wavelengths in space for each frequency.

The body, head, and pinnae all reflect some sound. The body and head reflect low and high frequencies more evenly, while pinnae redirect very high frequencies in patterns that are congruent with their ridges and folds. Sounds reflected off of the body make their way to a listener's ear canals leading to certain spectral colorations of sound that depend on the location of a sound source relative to a listener.

This body-reflection spectral coloration creates a cue which is considered monaural because the human audiospatial system can interpret such spectrum-modified sounds without needing to compare them with another ear's received sound. Spectral coloration differs for all sound source locations, so this cue can be used to discern vertical and horizontal plane locations. This cue is considered to be required for vertical sound localization.

Another monaural cue exists due to how the ear's pinna has directional selectivity for front-arriving sounds. Back-arriving sounds have some of their higher frequencies

reflected and absorbed by pinnae. This cue enables localization decisions for whether a sound emits from in front of or behind a listener. Both spectral coloration from wave/body interaction and the front/back discretion from pinnae are encoded into HRTFs.

Figure 2.3 shows pressure fields produced by a single-frequency point source with and without a body present. The sound pressure that would have been at the location of a listener’s right ear when there is no body present is regular and predictable in this case. However, adding a body creates disturbances for the sound pressure at the listener’s right ear due to reflections. Small movements of a listener’s body can change the characteristics of arriving sound pressure per frequency.

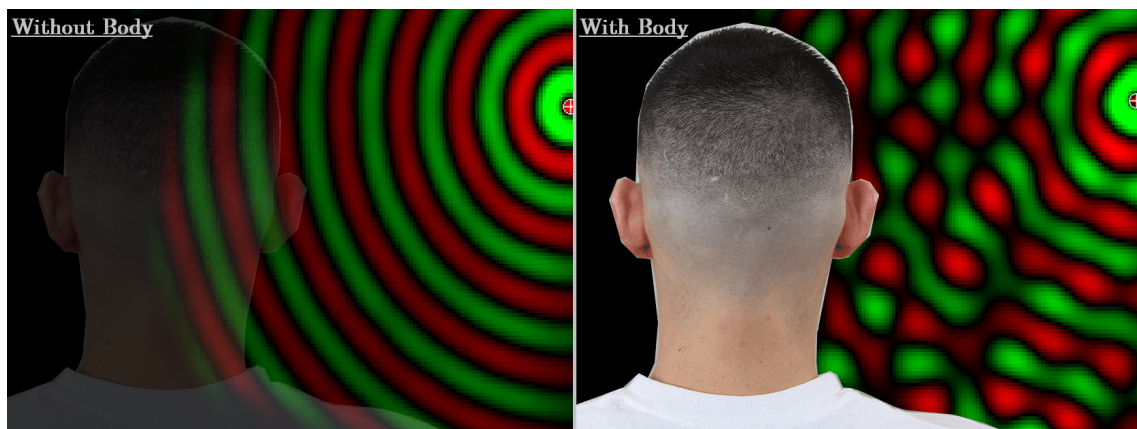


Figure 2.3: The resulting pressure field from a point source is shown without a body present (left) and with a body present (right). Green is high pressure, black is ambient pressure, and red is low pressure.

2.5 Apparent Source Size

It is possible for humans to perceive a sense of size for a sound source. This percept has been studied mainly as ‘apparent source width’ (Blauert, 1997; Potard and Burnett, 2004), but ‘apparent source height’ has also been investigated (Käsbach, 2016). Apparent source height is affected mainly by loudness and sound signal frequency content. However, apparent source width is affected by those two factors as well as a binaural factor called ‘interaural coherence’.

A measurement called ‘interaural cross-correlation’ is defined by de Vries et al. (2001) as the cross-correlation between near-term left and right ear signals which are normalized by their root-mean-square values, and where a time delay of ± 1 ms defines the range of cross-evaluation. Interaural coherence is calculated as the maximum value of a short time period’s interaural cross-correlation (Faller and Merimaa, 2004). Apparent source width increases as interaural coherence decreases.

The apparent effects of source size do not appear in HRTFs like the rest of the spatial cues mentioned so far. That is because HRTFs are defined for point source locations relative to a listener and not for extensive sources. However, by using

signal decorrelation or decomposition, multiple uncorrelated point sources can be spatialized so that HRTF spatial cues and control of interaural coherence can be used together to synthesize spatially extensive sound sources.

3 Time-Frequency Domain Audio Processing

‘Time-frequency domain signal processing’ is the processing of signals that contain both time and frequency information in their representations. With a finite-length time-domain signal, the time-value information is perfectly available, but frequency information is unknown without further calculations. And with a frequency-domain representation of that same signal, the frequency information is readily available, but then time information between the beginning and end of the signal is no longer at hand. To meet the requirements for signal processing of frequencies at arbitrary moments of time in a signal, various ‘time-frequency representations’ for signals have been devised (Chan, 1982; Delprat et al., 1992). A time-frequency representation is the use of frequency-representations at various points in time throughout a signal.

Time-frequency representations of signals lend themselves well to the study of acoustics and audio signal processing. A particular technique known as the ‘Short-Time Fourier Transform’ (STFT) has matured in audio signal processing. It has been applied for analysis of speech signals as early as 1970 (Oppenheim, 1970) and for analysis, modification, and synthesis of audio signals as early as 1976 (Callahan, 1977). STFT makes use of a frequency-transform known as the ‘Fourier transform’ by applying it across many short time windows of a larger time-domain signal. A Fourier transform decomposes a time-domain signal into sinusoidal components of different frequencies that would reconstruct the original time-domain signal when rendered all together.

With the steady increase in computing capabilities, real-time (i.e. online) processing with time-frequency representations of streamed time-domain signals has become a possibility (Boashash and Black, 1987).

This thesis makes use of real-time time-frequency domain (TF-domain) audio processing when decomposing time-intervals of mono signals into frequency components, and when synthesizing time-domain ambisonic B-format signals from those frequency decompositions after spatialization. That is why it is important to review these techniques here.

3.1 Short-Time Fourier Transform in Real-Time

The plug-in architecture for this implementation (more detail in Chapter 8) provides input blocks of audio samples for processing, and expects output blocks of audio samples to be filled. An input or output block contains one period of recent time-domain samples for various signal channels. An example input block could be: two input signal channels, 512 samples for each channel at a 48 kHz sampling rate which amounts to ~10 ms of audio for each channel. A block period is usually short enough to represent a recent localized time-region in a signal, and it is long enough to provide useful resolution after it has been transformed into a frequency-domain representation.

The STFT method was selected as the means for working with a TF representation

of the input and output signals, and it can be run in real-time for many audio channels.

3.2 Overlap-Add Short-Time Fourier Transform

This particular STFT method is an Overlap-Add (OLA) STFT. The first step in the OLA STFT procedure is to slice a time-domain signal into short, partially overlapping intervals. For instance: a time domain signal may be sliced into 50 ms intervals with each interval overlapping 25 ms with the previous interval in its first half and 25 ms with the later interval in its second half. The distance between the beginnings of the intervals is called the “hop size”. Next, each interval has a window applied to it so that its beginning and end fade in/out; this is done to shape spectral leakage, mitigate the effects of the discrete Fourier transform’s circular nature, and so that overlapping intervals don’t provide excess energy in overlapped sections. Then, the windowed intervals are transformed into frequency components via a ‘fast Fourier transform’; an efficient algorithm for calculating a discrete Fourier transform.

For one of these windowed intervals $\mathbf{x} = x_0, x_1, \dots, x_{N-1}$ of length N , applying the discrete Fourier transform to it will produce N frequency components $\mathbf{X} = X_0, X_1, \dots, X_{N-1}$. A frequency component X_k is produced by consideration of each time-domain sample x_n as

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}. \quad (1)$$

This form is known as the ‘forward discrete Fourier transform’. After applying the forward discrete Fourier transform to a recent windowed interval, the frequency-domain representation of that local time interval, \mathbf{X} , is considered a unit of a TF representation suitable for analysis and modification.

Sometimes nothing more than the analysis of frequency-domain components may be required. However, some applications which modify frequency-components or create frequency-domain signals will have the intention of resynthesizing time-domain signals. An array of frequency-components can be transformed into an interval of time-domain samples \mathbf{x} where each sample x_n is rendered as

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{\frac{i2\pi}{N}kn}. \quad (2)$$

This form is known as the ‘inverse discrete Fourier transform’. After applying the inverse discrete Fourier transform, an interval of time-domain samples is produced. By partially overlapping many intervals that were produced in this way, a TF representation in an STFT format can be stitched together to reconstruct full time-domain signals.

An example of analysis, modification, and resynthesis using the OLA STFT procedure is shown in Figure 3.1 in four steps. In step 1, an original time-domain signal is sliced into intervals of 1024 samples with a hop size of 512, and then they each

have a Hann window applied to them. In step 2, each windowed interval is transformed into the frequency domain with a forward discrete Fourier transform. Then in step 3, a low-pass filter is applied to each of the frequency domain representations of the windowed intervals by using an element-wise vector multiplication of each interval's frequency components being multiplied with the corresponding low-pass filter gains for each frequency. This low-pass filter starts rolling off at 500 Hz and attenuates 12 dB per octave. In step 4, each of the frequency representations for the windowed intervals are inverse discrete Fourier transformed back into time-domain representations and then summed into a final output signal according to their locations in time. The final output is a time-domain version of a TF-domain-modified signal.

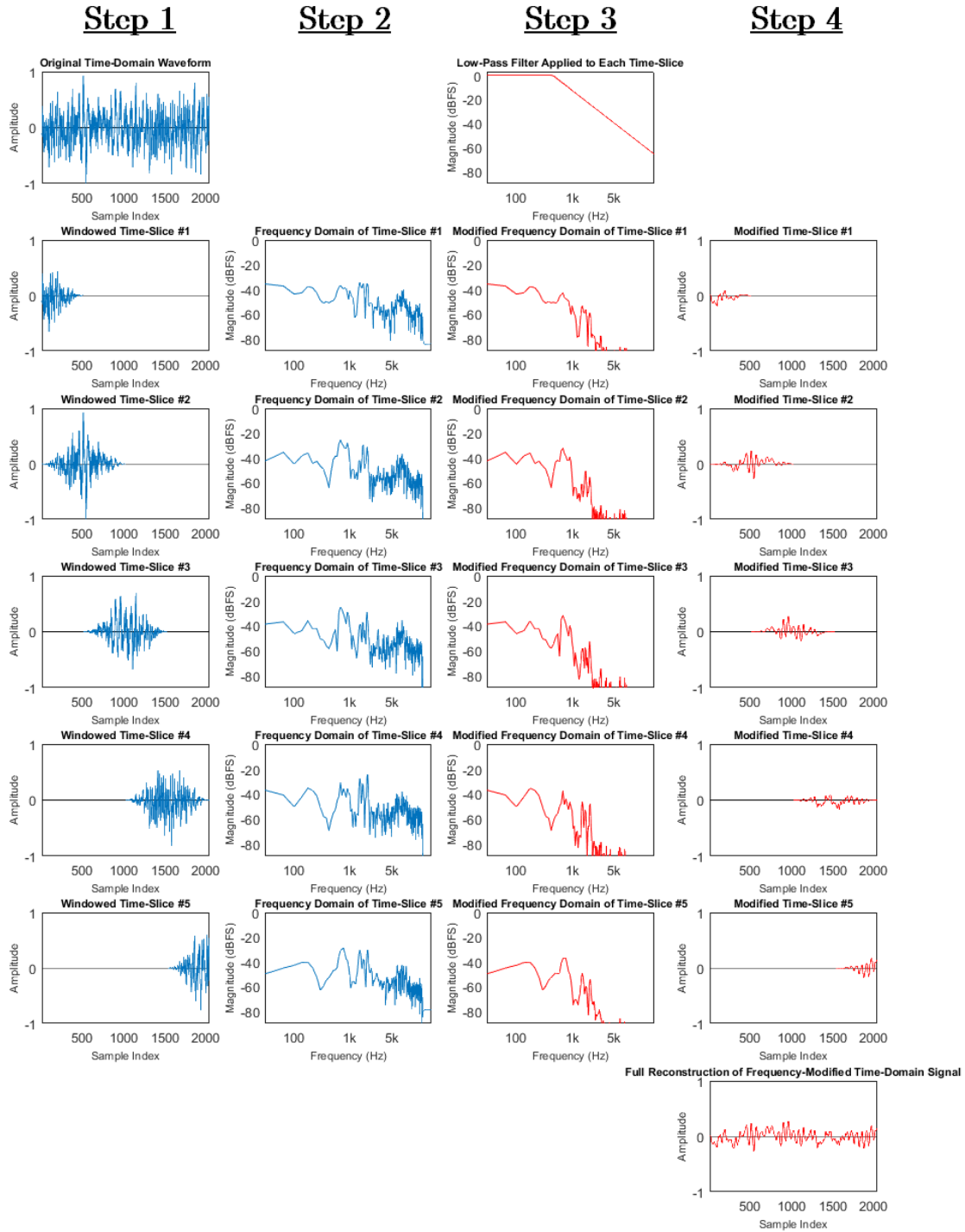


Figure 3.1: An illustration of the Overlap-Add Short-Time Fourier Transform (OLA STFT) procedure exhibiting analysis, modification, and resynthesis.

3.2.1 Alias-Free Short-Time Fourier Transform

Undesirable aliasing artifacts appear when frequency parameters are modified between hops and then each hop is resynthesized back into time-domain signals. This is caused by the circular nature of discrete Fourier transforms even with windowing before the original forward Fourier transforms.

An implementation of STFT that aims to remove this type of aliasing was used in this thesis work called ‘afSTFT’ (‘Alias-Free STFT’) (Vilkamo, 2015). The way in which this implementation avoids aliasing is by providing large, zero-padded buffers for each hop during analysis and resynthesis. The windowed segments of audio in each hop are placed into zeroed signals that are longer than the hop segments themselves. When they are resynthesized after frequency manipulations, then there is enough space in the resulting discrete signals so that any extensions to the original hop signal length can play out without reaching the ends of the signal. If any extensions in time were able to reach the ends of the signal, then they would wrap from the end to the beginning, or from the beginning to the end; this is the circular nature that is avoided by using afSTFT.

4 Spatial Audio

In order to induce the experience of hearing spatially extended sound events, we need an audio format that can carry spatial information and we need ways of reproducing spatial cues for listeners using either loudspeakers or headphones. In this section we will discuss ambisonics B-format which will be used as the carrier surround sound format for the results of this implementation’s synthesis. We will also discuss methods for reproducing sound fields or spatial cues from B-format signals with loudspeakers and headphones.

4.1 Ambisonics and B-Format

‘Ambisonics’ is a surround sound format that was developed in the 1970s. In efforts to encode and reproduce sound fields that pass through a sphere, Gerzon recognized that spatial audio signals have properties enabling orthogonality. Since audio signals can be orthogonal, then a spherical harmonic representation of orthogonal signals can be used to describe the sound field around a point in space (Gerzon, 1973). In his work, Gerzon described how truncated spherical harmonic decompositions of sound fields can be created from combinations of orthogonal signals, and how higher order representations that contain more orthogonal components will provide more accurate approximations of sound fields.

If spherical harmonics components were sought to be recorded directly from an acoustic space, then a microphone configuration would need to be arranged in a way that we call ‘B-format’. However, above first-order ambisonics the B-format configuration becomes impractical for recording due to the large number of coincident microphones directed at a single point in space. Luckily, alternative configurations are possible for producing spherical harmonic signals, since signals from multiple microphones can be rotated or linearly combined to form others. ‘A-format’ is a configuration where, for first-order ambisonics, four cardioid microphone capsules are arranged to point towards their own face of a tetrahedron. An example of such a microphone array is the SoundField SPS200 (shown in Figure 4.1). From A-format signals, B-format signals can be derived so that natural spherical harmonics processing can be done to determine the sound field surrounding a point. For higher-order ambisonics recording, one available solution is the Eigenmike EM32 which is a 32-capsule spherical microphone array that can have its inputs processed into fourth-order ambisonics B-format (Eigenmike EM32 shown in Figure 4.1).



Figure 4.1: SoundField SPS200 (left) and Eigenmike EM32 (right).

4.1.1 Ambisonic Panner

The incident sound of a virtual point source can be easily spatialized in ambisonics by first assuming that it is far enough away from a virtual B-format microphone so that its wavefronts are plane-like. Under the planar wavefront assumption, a virtual source’s signal can be incorporated into a truncated spherical harmonics representation of a sound field by calculating the gain factor that would be observed by each spherical harmonic component’s directional pattern given that there is a source placed at some relative azimuth and inclination. First-order B-format directional patterns are shown in Figure 4.2. For small spherical harmonic orders (typically $\leq 3^{rd}$ order), “gain tables” are often employed that contain simple functions of azimuth and inclination which calculate each orthogonal component’s gain (Schacher and Kocher, 2006). However, the gain factors for any spherical harmonic order are derivable through a process involving Legendre polynomials (Daniel, 2000).

The implementation presented in this thesis makes use of a small Spherical Harmonics library (SHlib) in order to calculate orthogonal component gains for spatialized point sources. SHlib uses the Legendre polynomial calculation for orthogonal spherical harmonic components for any order N .

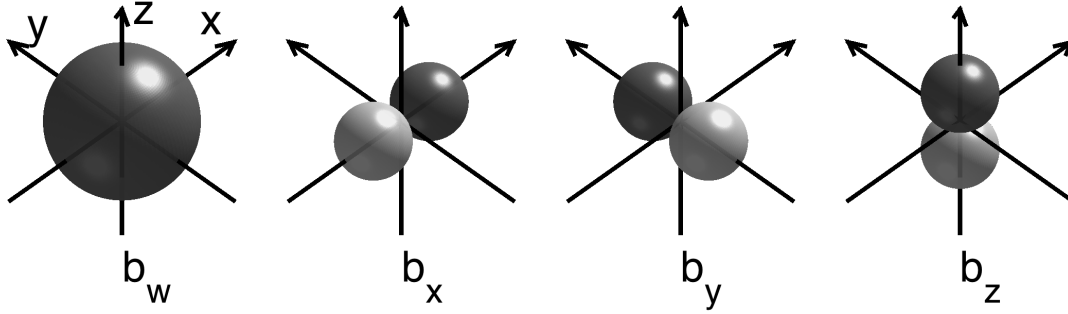


Figure 4.2: First-order ambisonics B-format components and their directional patterns. The lighter patterns indicate phase-inverted responses. Reprinted from ‘Parametric Time-Frequency Domain Spatial Audio’ (Pulkki et al., 2018). Reprinted with permission.

4.2 Directional Audio Coding (DirAC)

The spatial extent synthesizer presented in this paper assumes that there is some sufficient way of reproducing ambisonic B-format signals acoustically for listeners to hear. Pleasant and accurate ambisonic reproduction has been historically hard to achieve. For example, many published listening test results were made using suboptimal decoding (Heller et al., 2008).

One known suitable method is called ‘directional audio coding’ (DirAC) (Pulkki, 2007). At initialization, DirAC is given a spatial description of a loudspeaker configuration so that it can direct virtual cardioid microphones towards loudspeaker locations within incoming B-format signals. The virtual microphones produce signals that can play from each loudspeaker, but soon those signals are manipulated to reduce coincident frequencies from conflicting directions. For sounds that arrive directly to a listener position, it is ideal to have them reproduced as near to their apparent emanating direction as possible with few loudspeakers. However, for diffuse sounds that arrive from many indiscernible directions, it is best to produce the sounds in an uncorrelated manner. DirAC performs direction of arrival (DoA) estimation and diffuseness estimation by comparing prominent DoAs with the omnidirectional signal component in order to split an ambisonic signal into two streams, non-diffuse (direct) and diffuse. This step in DirAC is called the ‘energetic analysis’. It does this DoA estimation and diffuseness estimation for each frequency component when using an STFT approach, or for narrow frequency bands when using a filter-bank approach. Non-diffuse streams are decoded as point sources optimally panned between loudspeakers using ‘vector base amplitude panning’ (VBAP), while diffuse streams are decorrelated so they are heard to be randomly spread around the listening position. A block diagram for DirAC using a filter-bank approach is shown in Figure 4.3.

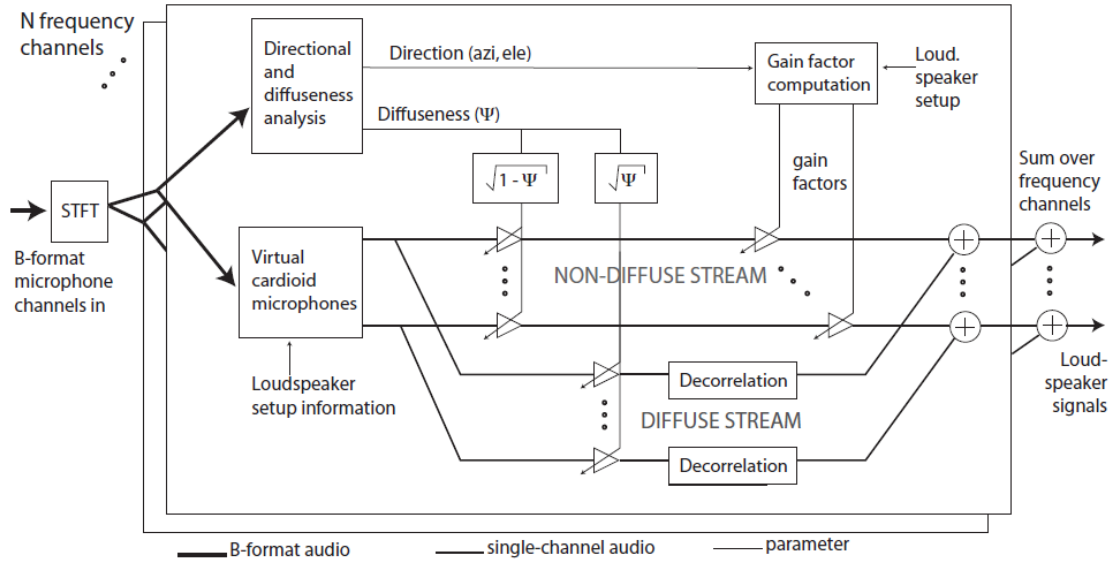


Figure 4.3: An incoming B-format signal is filtered into N frequency bands with STFT, virtual cardioid microphones produce naïve loudspeaker signals, directional and diffuseness analysis (energetic analysis) provides direction and diffuseness parameters, diffuseness parameter affects direct and diffuse stream gains, VBAP uses direction parameters to produce gain factors that affect the direct streams, direct and diffuse streams are summed for each loudspeaker, and then all N frequency bands are recombined to produce final loudspeaker signals. Reprinted from Ville Pulkki’s slides for the course ‘Communication Acoustics’. Reprinted with permission.

This description of DirAC is based off of material from the introductory years of the method. DirAC has seen marked improvements over the years since. More detail can be found in the book *Parametric Time-Frequency Domain Spatial Audio* (Pulkki et al., 2018).

4.3 Binaural HRTF Audio

The linear transformations that happen to sounds before they arrive at a listener’s ear canals, including the spatial cues mentioned in Chapter 2, can be recorded through various means of measurement (Gardner et al., 1994; Algazi et al., 2001). The spatial cues that occur from direct sound arriving at a listener can be synthesized for binaural headphone reproduction by using a database of linear filters for left and right ears. The database of filters must be addressable by the azimuth and elevation (and sometimes distance) that a sound source emits from relative to a listener’s orientation. The filters in such a database are called ‘head-related transfer functions’ (HRTFs) when they are represented as functions of frequency, and they are called ‘head-related impulse responses’ (HRIRs) when they are represented as functions of time. Spatializing a point source into binaural audio can be done by convolving a source audio signal with left and right ear HRIRs/HRTFs for the corresponding

relative azimuth and elevation (producing two output signals from just one input signal).

The process of spatializing point sources for binaural headphone reproduction has identical inputs when compared to the basic ambisonic panner explained in 4.2.1; it takes as input a source audio signal and polar coordinates of azimuth and elevation.

4.4 Target Audio Encoding for This Thesis

The three primary spatial hearing cues that were described in Chapter 2 each depend on the direction of arriving acoustic wavefronts and the physical interaction between a listener’s body and those arriving wavefronts. If we were to encode this implementation’s synthesized audio for binaural headphone listening by applying HRTFs, then we would not be able to appropriately reproduce a sound field with a loudspeaker setup. There would already be spatial hearing cues embedded into the stereo audio. Surround sound loudspeaker reproduction of spatial audio prefers that physical interactions are played out in a listener’s personal space so that spatial hearing cues are naturally produced.

Synthesizing spatial audio into a format that carries information about the sound field near a listener while not containing spatial hearing cues would be ideal if an HRTF effect could be applied later to synthesize spatial hearing cues optionally. Thankfully, ambisonics B-format is a suitable audio format for this. It has been shown that binaural signals can be produced from B-format signals (Noisternig et al., 2003; Politis et al., 2017).

B-format is chosen as the synthesized audio format for this implementation since it can be used in loudspeaker and headphone reproduction. DirAC and various ambisonic decoding methods exist for use in loudspeaker sound field reproduction. Just as well, binaural renderers are available that use B-format signals as input. A particular implementation of a DirAC-based binaural renderer for B-format input signals was used to field development of this thesis, and it was eventually embedded into the Unity game engine plug-in that was developed (Politis et al., 2017).

5 Method for Spatial Extent Synthesis

This thesis' main method for spatial extent synthesis relies on the concepts explained in Chapter 3 and Chapter 4. Drawing from Chapter 3, at the beginning of the extent synthesis there are forward discrete Fourier transforms that decompose each mono signal into its frequency components. At the end of the spatial extent synthesis, frequency distributions for each B-format channel will be inverse discrete Fourier transformed to create a time-domain B-format signal. Related to the contents of Chapter 4, locations of point sources will determine the amount of gain applied to their signals as they are mixed into B-format channels.

Recall that spatialization of a single point source can be performed by way of an ambisonic panner which was introduced in Chapter 4 (see 4.1.1). Spatial extent synthesis can be accomplished by decomposing a mono signal into N independent “component” signals, and then spatializing those N signals as if they were their own point sources. When those N point sources (also called “sound vertices” or “component-point sources”) are distributed in a way that they are evenly upon or within a virtual object’s geometry, then the anechoic extent that is audibly perceivable will be a physically accurate projection from the virtual object to the listener. As the virtual object or the listener change orientations, the apparent spatial extent will be updated to reflect the virtual-world state.

In Figure 5.1, a single point source is shown with a line directly connecting it to a listener position. In Figure 5.2, 128 sound vertices representing frequency components of a mono signal are shown connected to a listener position.

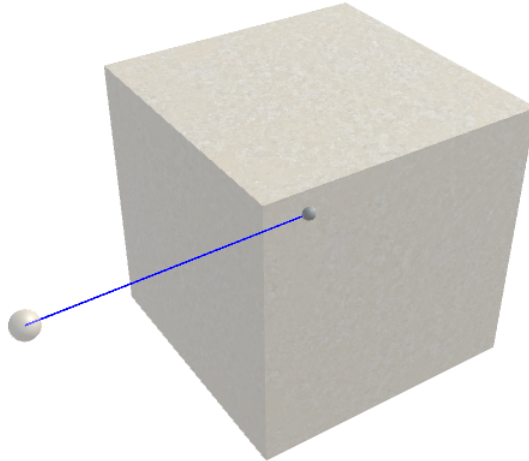


Figure 5.1: A single point source at the center of a cube projects towards a listener position depicted as the sphere on the left.

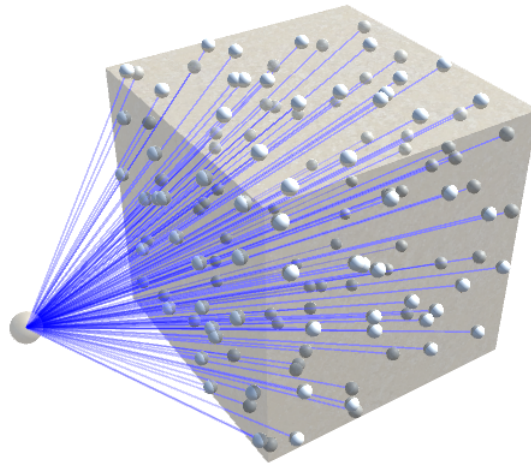


Figure 5.2: 128 sound vertices (or component-point sources) are distributed across the surface of a cube, one for each frequency component, and their projections towards a listener position are depicted as blue lines converging to the sphere on the left.

Comparing figures 5.1 and 5.2, it can be seen that one point source is insufficient for a sizeable sound source object such as the cube that is shown, while 128 component-points distributed across the cube geometry provides spatial extent that is no greater than the largest apparent hull of the object.

5.1 Spatial Extent Synthesis Steps

This section details the step-by-step process for spatial extent synthesis that was implemented.

5.1.1 Geometry Initialization

Before a spatially extended virtual source can be rendered in real-time, it first has to have its geometry initialized. The purpose of initialization is to uniformly distribute frequency components as points across a virtual source's specified geometry. The methods for distributing points across various 3D primitive geometry and arbitrary 3D meshes is detailed in Chapter 6.

Occasional initialization happens in two steps:

1. The programmer or program specifies a shape (geometry) for a sound source.
2. On the first rendering step after a source shape change, points are distributed across the geometry as though it is in its default orientation, scale and location; in other words, no rotation is applied, scale is (1, 1, 1), and the geometry

originates from point $(0, 0, 0)$. The number of points is hardcoded to 128, but this number could be allowed to change. The points are stored in an array called “ShapePointsOriginal”.

5.1.2 Scale, Rotation, Translation

On the first render after a ‘Geometry Initialization’, or when a source’s orientation, scale, or location have changed, then a ‘Scale, Rotation, Translation’ step must be performed.

‘Scale, Rotation, Translation’ occurs in the order of its name:

1. **Scale** – The source’s points in “ShapePointsOriginal” are scaled in X, Y, and Z dimensions according to the source’s shape scale settings. The results are stored in an array called “ShapePoints” which will hold the current shape points to be used for render later on.
2. **Rotation** – The source’s points in “ShapePoints” are rotated according to an axis-angle representation of a rotation. When the programmer specifies a source’s orientation, they provide a unit vector that specifies an axis for which to rotate around and an angle for the amount of rotation that should be applied around that axis. The position after rotation for each point is calculated using Rodrigues’ rotation formula (see Appendix A). Results are once again stored into “ShapePoints”.
3. **Translate** – The source’s points are lastly translated to the location of the source object by adding the source position to each point in “ShapePoints”.

Now the points in “ShapePoints” are ready to be used for spatial extent synthesis.

5.1.3 Frequency Decomposition, Component Spatialization, and Mixing

Each spatially extended sound source now has its “ShapePoints” array prepared. That means we are ready to place frequency components at each point for every source and then mix them into a frequency-domain B-format signal by use of ambisonic panning. At the beginning of this render step, a final-mix frequency-domain B-format signal is cleared so that each complex frequency component is $(0 + 0i)$.

For each sound source:

1. The sound source’s mono signal in this render period’s input audio block is transformed to the frequency domain with a forward discrete Fourier transform. In other words, frequency decomposition is applied to the sound source.
2. Each frequency component is spatialized by placing the frequency component at its corresponding point from “ShapePoints” and then performing ambisonic

panning as described in Chapter 4. Frequency components and shape points follow a 1-to-1 mapping in order. Specifically, a frequency component's gains to be applied for each channel in the B-format signal are calculated based on its point's location relative to the listener position, and then the frequency component is multiplied by each gain value and added to the corresponding B-format channels at the same frequency bin index for which the frequency component belongs.

After this process has been applied for all sound sources, a complex frequency-domain B-format signal will contain the mixed result of all sound sources' spatial extent synthesis.

5.1.4 Inverse Fourier Transform for Final-Mix B-Format Signal

The rendering process for one block of time is coming to its end. The desired output of this spatial extent synthesis for multiple sources is a single time-domain B-format signal.

For each channel in the frequency-domain B-format signal:

- Apply an inverse discrete Fourier transform to the frequency-domain channel signal and write the time-domain result into the corresponding channel in the final-mix output time-domain B-format signal.

5.1.5 Soundfield Reproduction from Final-Mix B-Format Signal

Finally, the synthesized time-domain B-format signal is ready for storage or decoding. It can be decoded by DirAC or various ambisonics decoders for soundfield reproduction with loudspeaker setups or for binaural playback with headphones.

5.2 Alternative Methods for Signal Decomposition

This procedure for spatial extent synthesis can have its decomposition and resynthesis methods substituted. Any process which can produce multiple uncorrelated components from a mono sound source signal has promise for use in pointwise spatial extent synthesis. That is because when uncorrelated components are spatialized through ambisonic panning and sound emits towards a listener from sufficiently sparse locations, then the conditions for inducing apparent source size perceptions larger than point sources are produced (mainly lower interaural coherence).

Another viable time-frequency domain method is the use of the 'discrete sine transform' (DST) or the 'discrete cosine transform' (DCT) and their inverses at each interval in a TF-domain process. A form of this spatial extent synthesis using DST was experimented with in MATLAB and the artifacts that were produced were less bothersome in some respects than those that appear when using Fourier transforms.

A form of time-domain decorrelation using granular synthesis was attempted in experiments. Granular synthesis can provide any number of components for spatialization by shortly windowing an original mono sound signal many times at sample offsets so that it can be recomposed by the sum of many small “grains”. This form of time-domain granular decomposition and resynthesis was experimented with in MATLAB and in the real-time implementation, but undesirable artifacts were always present in the simple attempts that were made.

Forms of time-domain decorrelation that have found success in auditory displays are ‘full band decorrelation’ and ‘dynamic decorrelation’ which use all-pass filters with random, noise-like phase responses to create multiple perceptually equal yet statistically uncorrelated signals (Potard and Burnett, 2004). Dynamic decorrelation is similar to full band decorrelation, but the phase response for the all-pass filters varies in time so more uncorrelated components can be generated.

6 Point Distribution onto 3D Geometry

When we seek to spatially extend a mono signal, we need to decide how the region of extension will be determined. Manual control of spatial extent is common in audio mixing for music and video purposes, but virtual reality applications require automatic and realistic decisions for creating spatial extent. This thesis aims to expand the possibilities for automatic audio spatial extension in virtual world environments by allowing designers to spatialize arbitrary 3D meshes. Also, additional 3D primitive shapes and a point cloud shape are implemented.

The presented method for spatially extending sound given various types of geometry requires low-discrepancy distribution of points across geometry surfaces. This means it is important to distribute a small set of points as evenly as possible across a surface. To accomplish even distribution, low-discrepancy Halton sequences are applied similar to how they were used in Pihlajamäki et al. (2014). This thesis focuses on distributing decomposed mono signals across shape surfaces rather than throughout shape volumes. This is because solid and opaque sound objects became of interest during this research. Later in this chapter, the pros and cons of surface distributions are discussed, and an experimental type of extent distribution is described for opaque sound objects.

6.1 Halton Sequences

‘Halton sequences’ (Halton, 1964) are deterministic, low-discrepancy sequences that appear random. “Low-discrepancy” means that for any length of the sequence that starts from the beginning, the set of elements will have a quality where the collection of distances of members to their nearest neighbor has low variance. The fact that Halton sequences are deterministic is useful for achieving identical spatial extent distribution for identical geometry during any execution of this implementation. Finally, how Halton sequences appear random is useful so that no strong patterns appear in the audio of extent distributed signal decomposition components.

Uniform random distributions created from uniformly distributed random samples are less suitable for extent distribution because they are not deterministic and there are no guarantees for low discrepancy. Since they are non-deterministic, every instance of the same geometry would have different extent distributions unless they were pseudorandom and initialized with the same seed. And since uniform random distributions are not low discrepancy, then there is likely to be bunching of points or voids for small sample sets.

The benefits of Halton sequences over uniform random samples is obvious when illustrated. In Figure 6.1, 128 points are shown after distribution with a 1-D Halton sequence versus 128 points distributed using uniform random samples.

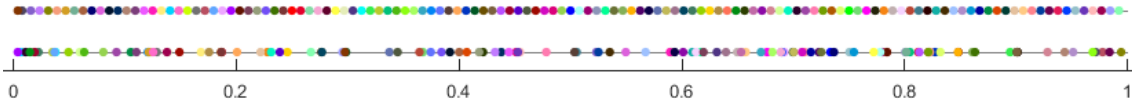


Figure 6.1: 128 points distributed with a Halton sequence (top). 128 points distributed using uniform random samples (bottom).

Halton sequences are created by using a base, usually 2 or 3, which divides the interval $(0, 1)$. First, every fraction of the interval $(0, 1)$ is calculated for the base. Then the base is raised to the next highest power and division continues once again until all fractions are calculated. This process repeats while the order of fractions calculated for each power of the base are staggered.

For a sequence with base 2, the interval $(0, 1)$ is divided in half to produce the first value in the sequence: $\frac{1}{2}$. Then since there are no other fractions with a divisor of 2 within $(0, 1)$, the base is raised to the next power and becomes $2^2 = 4$. Then the fractions $\frac{1}{4}$ and $\frac{3}{4}$ are calculated and appended to the sequence, and the base is raised again to $2^3 = 8$. The sequence of the fractions for each power of the base are spread optimally so they will not appear in order once the base raised to its current power is greater than 2. This carries on ad infinitum. The beginnings of the Halton sequences with base 2 and base 3 are

$$\text{Base 2: } \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \dots$$

$$\text{Base 3: } \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \dots$$

For quasi-random sampling of \mathbb{R}^n , n coprime bases are often chosen for n Halton sequences so that each sequence has low correlation with every other sequence. For \mathbb{R}^2 , a sequence with base 2 and a sequence with base 3 are typically used.

6.2 Primitive Shapes

‘Primitive shapes’ are simple, mathematically defined shapes. This means that computing surface distributions and calculating distances to shape surfaces have direct forms that are quick to resolve. These primitive shapes are useful for creating environmental sound effects, regional ambient sound sources, and approximating some simple convex 3D objects.

As was mentioned in this paper’s introduction, earlier work done with VW-DirAC provided spatial extension for cuboid, sphere, and cylinder shapes volumetrically. It also provided surface distribution for spheres and cylinders when projecting B-format signals onto those shapes (Pihlajamäki and Pulkki, 2012). We can investigate the design of a new primitive shape ‘hemisphere’ to demonstrate primitive shape surface distribution and distance to surface calculations.

Hemispheres are constructed similar to spheres. However, hemispheres can provide a sense of ground beneath a listener when making ambient sounds since sound will only come from above the ground plane. Low-discrepancy, or quasi-random, surface distributions require only two random variables because surfaces are areas that are parameterizable in two dimensions. However, points that are distributed across surfaces must still be three dimensional, so surface distribution requires that we transform two dimensions into three. Given two quasi-random variables, θ and z where $0 \leq \theta \leq 2\pi$ and $0 \leq z \leq 1$, a formula for picking points from the surface of a hemisphere is

$$\mathbf{p} = (p_x, p_y, p_z) = \left(\sqrt{1 - z^2} \cos \theta, \sqrt{1 - z^2} \sin \theta, z \right), \quad (3)$$

where θ is made from a Halton sequence where ordinary samples within the interval $(0, 1)$ are scaled by 2π , and z can just be another ordinary Halton sequence. The result of quasi-randomly sampling a hemisphere surface can be seen in Figure 6.2.

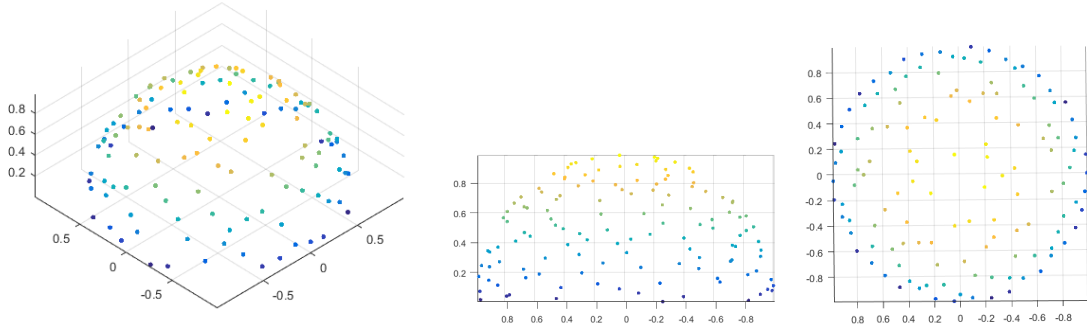


Figure 6.2: Hemisphere point-surface distribution shown at an angle and from the side and top.

To calculate the distance from a listener position to a hemisphere's surface (excluding its bottom), an axis check is first performed. If the listener position has a z -value greater than or equal to zero, then the calculation for the distance to the surface of a sphere can be used. If not, then the distance to the circumference of the hemisphere that lies on the x - y plane must be calculated. However, these calculations are only valid for non-scaled hemispheres. The nearest surface-distributed point to the listener position makes a good approximation for the distance to the hemisphere surface when scaling has been applied.

For the distance to a sphere surface, calculate the distance between the listener position and the sphere origin and then subtract the sphere radius. The absolute value of the resulting value will be the distance to the sphere surface. For the distance to a hemisphere circumference in the x - y plane, a point on the hemisphere circumference that is on the line drawn between the x - y flattened listener position (the listener position with its z -coordinate set to 0) and the hemisphere origin will be the closest point to the listener position. Finding that point and then the distance between the 3D listener position and that point results in the distance to the hemisphere surface for when the listener position has a z -value below zero.

6.3 Area-Based Application of Halton Sequences for 3D Mesh Sampling

A method for uniformly distributing points across the total surface area of an arbitrary 3D mesh was developed during this research, but it was discovered later that the same method, sans quasi-random variables, was already implemented for computer graphics research (Osada et al., 2002). A 3D mesh is most often defined by a set of triangle faces, but quads and higher order polygons also find use in meshes for computer graphics applications. This method has only been implemented to work with triangular meshes, but a similar approach could be taken for other polygons.

Broadly, the approach is to use a single Halton sequence to provide uniform samples which are then used to choose from a binned distribution of triangle faces. When a triangle face is chosen, then a pair of two more Halton sequences is used to place a point randomly onto the triangle face. There is one bin for each triangle face, and the probability that a bin is chosen depends on the ratio of the triangle face's area relative to the total mesh surface area. Each face has its own distinct starting point in the pair of coprime-based Halton sequences so to suppress patterns in a 3D mesh such as periodically spaced faces.

The procedure is as follows:

Setup

1. Create three Halton sequences. One of base 2, another of base 2, and one of base 3. The last two sequences are a coprime pair.
2. Create a list of integers, one for each face, starting at 0 each. This is for tracking how many points have been placed in a face; the "placement count".
3. Create an empty list for storing a running sum of mesh-face areas, and create a floating point number starting at 0.0 to store the cumulative surface area.
4. For each triangle face in the order that the mesh specifies, calculate the area with Heron's formula (see Appendix A). Add the triangle face's area into the current cumulative area, then append the current cumulative area to the end of the list.
5. After step 4 is completed for all triangle faces, then the list will contain a monotonically increasing sequence of the triangle faces' running sum in order and the cumulative area variable will represent the total area.

Point Distribution

1. For every point that is to be spatially distributed, take a new sample from the first Halton sequence to get a number within $(0, 1)$, then multiply that number by the cumulative surface area of the mesh. This results in an area-scaled sample.
2. Perform a binary search through the running-sum list to find the index where the area-scaled sample is less than or equal to the current value and greater than the previous value. That same index is used to address the list of triangle faces so that a point can be placed within the corresponding face.
3. After retrieving the face which will receive a point, use the sum of the placement count for that face and that face's global offset in the coprime Halton sequence pair to gather two quasi-random samples, r_1 and r_2 . Increment the placement count for the triangle face.
4. Use the pair of quasi-random samples (r_1, r_2) , and the triangle face vertices \mathbf{v}_A , \mathbf{v}_B , and \mathbf{v}_C to spatialize a point into three dimensions within the face's bounds. The 3D point, \mathbf{p} , is computed by

$$\mathbf{p} = (1 - \sqrt{r_1}) \mathbf{v}_A + \sqrt{r_1} (1 - r_2) \mathbf{v}_B + r_2 \sqrt{r_1} \mathbf{v}_C. \quad (4)$$

The results of this 3D mesh surface distribution of points can be seen in the following figures. In Figure 6.3, an ordinary wireframe of a car model's 3D mesh is shown from its side, top, and front. In Figure 6.4, the same wireframe is shown with small spheres added at the locations of points that were distributed across the mesh surface. In Figure 6.5, the car model is shown with a transparent shading and the same small spheres after surface distribution.

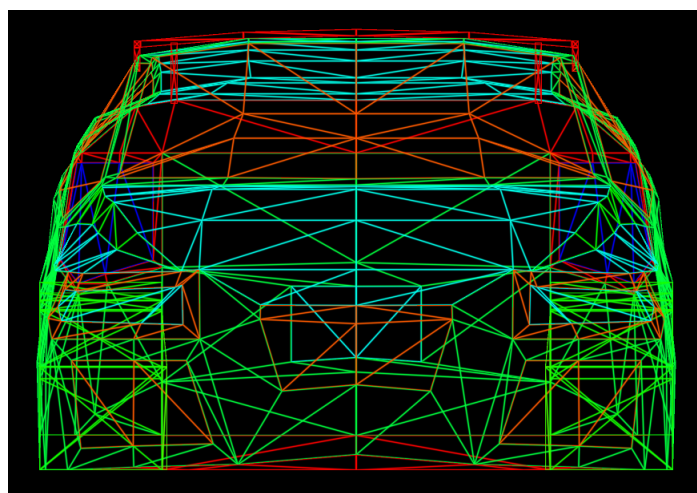
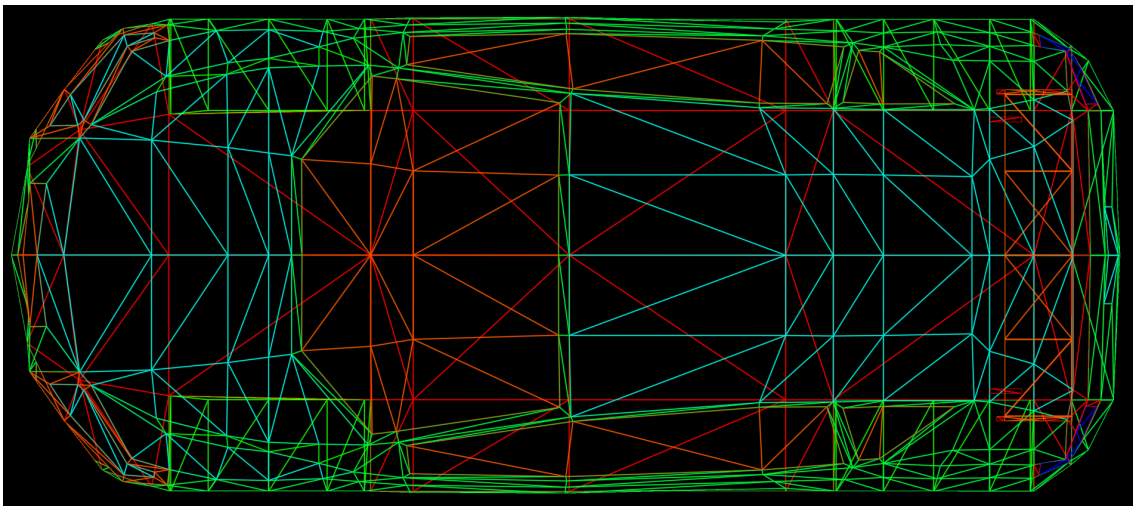
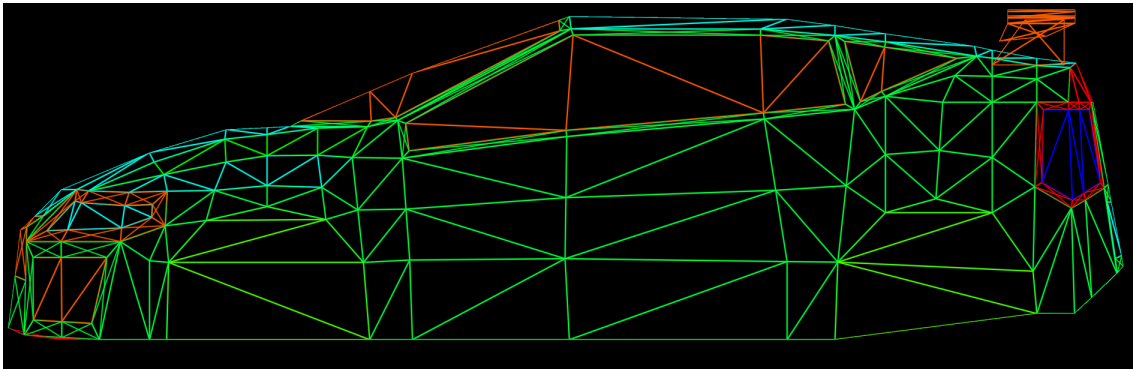


Figure 6.3: A wireframe display of a car model's 3D mesh. Perspectives in order: side, top, front.

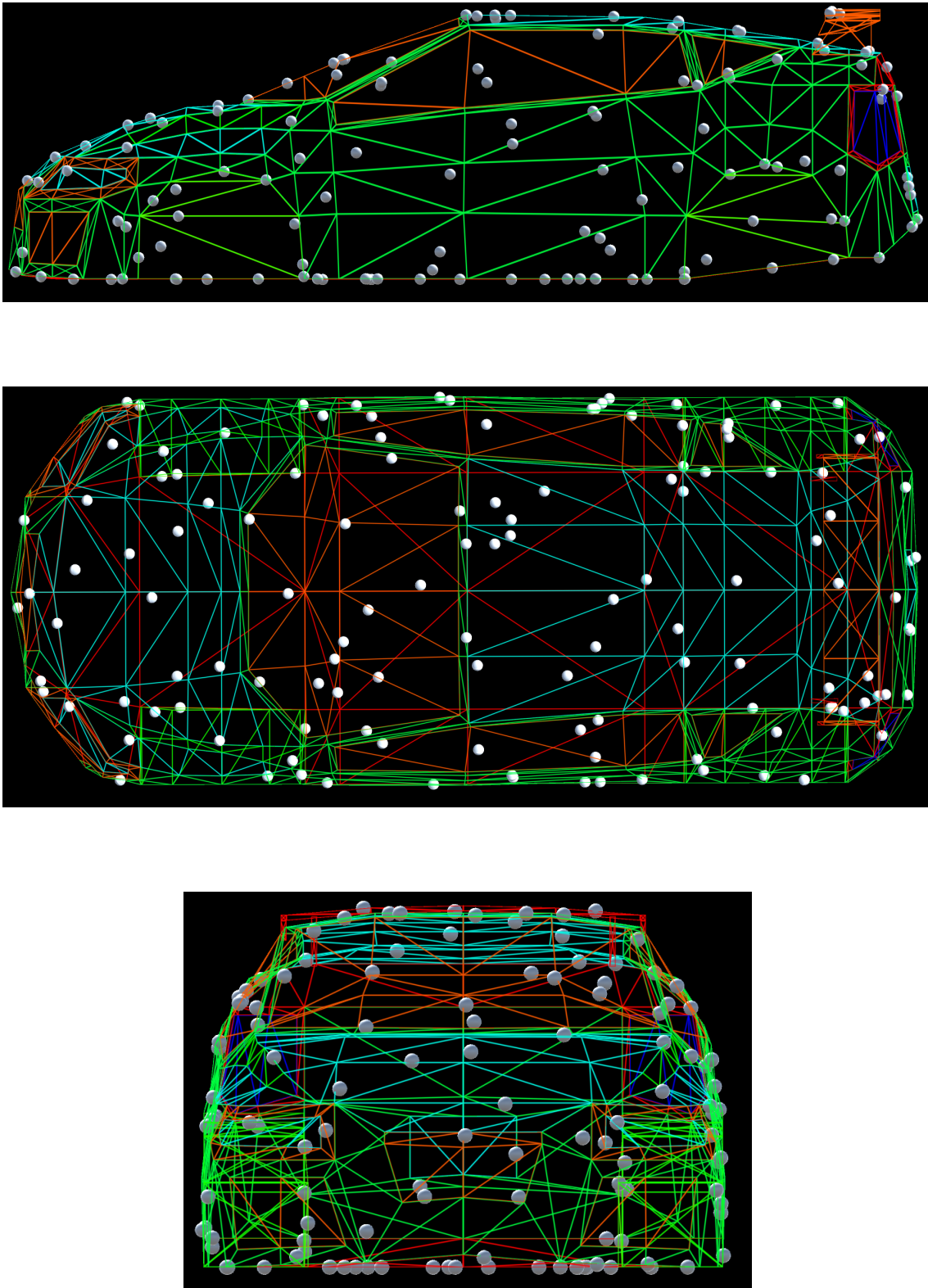


Figure 6.4: A wireframe display of a car model's 3D mesh with small spheres showing the locations of quasi-random point distribution across the mesh surface. Perspectives in order: side, top, front.

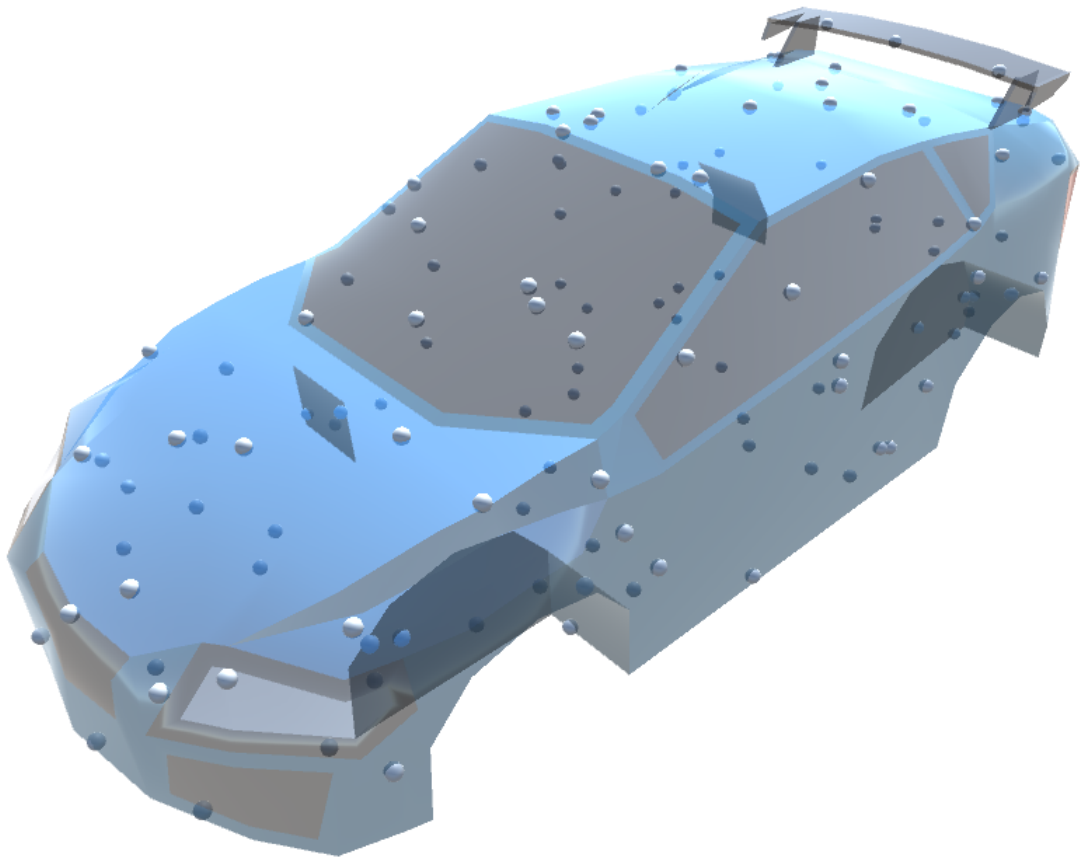


Figure 6.5: A transparent-shading of a car model with small spheres showing the locations of quasi-random point distribution.

6.4 Strengths and Weaknesses of Surface Distribution

This implementation deviates from the spatial extent synthesis found in VW-DirAC in that only surface distribution of points across shapes is supported. Although VW-DirAC supported surface distribution in addition to volume distribution, the surface distribution was only for projecting already spatially extensive B-format signals (Pihlajamäki and Pulkki, 2012).

Surface distribution has various pros and cons when compared to volumetric distribution for spatial extent synthesis of mono sound signals.

Pros

- Surface distribution is easier to compute for arbitrary geometry
- Surface distribution more tightly mimics the form of geometry when the number of points to be distributed is the same as what is used during volumetric distribution
- Surface distribution is immediately suitable for back-face and occlusion culling applications for creating solid or “opaque” sound objects where sound should appear to emanate from object surfaces

Cons

- Without culling hidden surfaces, surface distribution exhibits issues with spatial density of loudness for large objects even more so than volumetric distribution
- Surface distribution cannot provide cloud-like regions that the listener can travel through while volumetric distribution can

Although the limits of spatial extent are better approximated with surface distribution of sound vertices, the sound vertex density within the generated extent is more prone to density issues. When a line which connects a geometry’s surface to the listener position lies on the plane defined by the geometry surface, then sound vertices on that surface appear to the listener on a thin line. Furthermore, without additional processing, there is no culling of hidden surfaces being performed so then back-facing surfaces or surfaces that would naturally be occluded by nearer surfaces will appear with higher densities in the synthesized extent due to their greater distance from the listener. These varying densities of loudness in the spatial distribution change when the listener orientation or sound source objects’ orientations change. These changes may highlight density irregularities for listeners.

Volumetric distribution of sound vertices reduces the variance of local loudness densities, but such distribution for 3D meshes would require voxelization and there would still be issues that arise with low volume/high surface-area geometries and concave geometries. For example, distributing sound vertices within ribbon- and hook-like shapes would be difficult to implement and there would still be the same loudness density issues.

6.5 Opaque Sound Sources with Back-Face and Occlusion Culling

A modification to surface distributed signal components was experimented with that simulates solid or “opaque” sound objects. For anechoic sound, an opaque sound source should appear to have sound emanating from only its apparent hull, and it should self-occlude its sound coming from hidden surfaces. In order to achieve this effect, signal components that are distributed on hidden surfaces must be redistributed to the locations of apparent components.

‘Occlusion culling’ is a process found in 3D graphics rendering that reduces the number of mesh-faces that will be sent to a renderer for rasterization. ‘Back-face culling’ is another simpler process for removing faces of a 3D mesh from being rasterized, but it has less strict criteria. For convex-shaped meshes, back-face culling excludes much of the same mesh-faces that occlusion culling would also remove. In practice, back-face culling is usually performed before occlusion culling in order to reduce the cost of the latter operation. Back-face culling is very simple to implement compared to occlusion culling, so it was used for experimentation in creating opaque sound sources.

In 3D graphics rendering, faces are typically visible from only one of their two sides. The side which should be visible is determined by a face’s surface-normal vector, which is calculated from a face’s vertex winding (clockwise in some renderers, counter-clockwise in others). The goal of back-face culling is to remove mesh-faces from the rendering pipeline that would be rendered showing the wrong side. A face should not be rendered when its surface normal is oriented along the same direction as the observer-to-face vector, when that face’s surface-normal is orthogonal to the observer-to-face vector, or when its surface normal is oriented anywhere between those two former cases.

In this implementation, signal components that are distributed across mesh geometries have their hosting surface’s normal vector stored with them so that back-face culling can be performed. If a signal component is culled because its hosting surface is back-facing, then the signal component is temporarily placed at the position of another “visible” signal component. Each signal component has an order of preference for which other component positions it would like to be placed upon. The collection of these orderings for all components’ preferences was designed to limit imbalances in the distribution of culled components onto the set of apparent components.

An illustration of back-face culling’s effects is shown in Figure 6.6. In the top image, component-point surface distribution is shown as normal. In the bottom image, back-facing components have been removed. The components that were removed in the bottom image would have their contents relocated to the remaining apparent points.



Figure 6.6: A 3D model representing a car is shown with component-point sources distributed across its surface (top) and with some of the component-point sources removed due to back-face culling (bottom).

A naïve implementation of temporary component repositioning leads to drastic spatial artifacts at the moments when components become culled or re-apparent. To suppress these artifacts, a fading gain factor is calculated for each component so that a mixture of two similar components with different spatial locations is created: one from a component's original culled position and one from its temporary apparent position. A component's original position contribution has a gain $g_{\text{OP}} = 0$ when the component's hosting surface normal is exactly back-facing (parallel with the observer-to-component vector), and the original position contribution has a gain $g_{\text{OP}} = 1$ when the component's hosting-surface normal is exactly orthogonal to the observer-to-component vector. When a component is apparent, its gain is $g_{\text{OP}} = 1$, but when it is culled, then it has a gain $0 \leq g_{\text{OP}} \leq 1$. The succinct expression of this gain calculation is

$$\alpha = \hat{\mathbf{N}}_{\text{hosting surface}} \cdot \overline{\hat{\mathbf{OC}}} \quad (5)$$

$$g_{\text{OP}} = \begin{cases} 1 - \alpha, & \alpha > 0 \\ 1, & \alpha \leq 0 \end{cases}, \quad (6)$$

where $\hat{\mathbf{N}}_{\text{hosting surface}}$ is the hosting surface's normal unit vector, and $\overline{\hat{\mathbf{OC}}}$ is the origin-to-component unit vector. A culled component's temporary apparent position has a contribution of gain $g_{\text{TP}} = (1 - g_{\text{OP}})$. ($g_{\text{OP}} + g_{\text{TP}} = 1$) is always true.

This attempt at simulating opaque sound sources showed some promise for improving loudness distribution in informal listening tests. However, the added benefits came at the cost of additional spatial artifacts that appear during movement of either the listener or sound sources.

7 Distance Attenuation for Spatially Extended Sources in Virtual Environments

For previous research’s listening tests, it was beneficial that spatially extended sound sources and listeners remained stationary so that test results would remain more consistent. If a listener becomes familiar with a sound source, then once the distance between a sound source and the listener changes it becomes apparent that changes in distance attenuation have an effect on sound localization. Much research has gone into sound distance perception involving distance attenuation (Holt and Thurlow, 1969; Zahorik, 2002; Zahorik et al., 2005).

For distance attenuation, the following four methods have been included in this implementation:

1. Component-pointwise attenuation

Distance attenuation is calculated for all sound vertices relative to the listener position. This method enables a sense of depth perception at the expense of changes to the frequency magnitude distribution of the source signal. However, this technique creates a comb filtering sound when individual frequency components become louder or softer as either the listener or sound source change orientations.

2. Centroid attenuation

Distance attenuation is calculated from the origin of the 3D geometry and applied as the gain to all frequency components. This method maintains the frequency magnitude distribution of the source signal. It is suitable for immersive sound environments without clear geometric borders like crowds or insect swarms. This method loses effectiveness when a source’s shape points are significantly skewed or offset from the source position.

3. Nearest sound vertex attenuation

For a sound source, the sound vertex which is closest to the listener is used to calculate distance attenuation. The resulting attenuation is applied as the gain to all frequency components. This method also maintains the frequency magnitude distribution of the source signal. An encouraging quality of this method is that the surface of some 3D geometry can be sensed through movement of either the listener or sound source. This method is well suited for geometries that the listener will not enter, but it can also be used to simulate a distribution of point sources surrounding the listener which the listener can move towards for inspection.

There is one major drawback for this method where very sparsely distributed sound vertices will make perceptible gain changes when a listener moves past a surface. The deviations in gain can be removed by using the following method for well-behaving geometry.

4. Nearest surface attenuation

Finding the shortest distance to the surface of a shape brings the most accurate results. For analytical geometries like spheres, cubes, and tubes, this distance can be computed directly. However for arbitrary 3D meshes, without optimizations each triangle face of the mesh must have its shortest distance calculated and compared so the overall minimum distance can be found. Although three vertices for a tri-face define an infinite plane and finding the shortest distance from a point to such a plane is simple, finding the shortest distance from a point to a finite plane is more difficult to implement. The algorithm for deciding the shortest distance from a point to a finite plane requires conditional region testing, because a point may be closest to the inside of the finite surface, one of the surface edges, or one of the vertices that define the finite surface.

In informal listening tests, the nearest surface attenuation method provided the most convincing attenuation for solid objects with significant spatial extent. Following are visualizations of spatial extent and nearest surface distance attenuation.

In Figure 7.1, a 3D mesh that has had sound vertices distributed on its surface is heard from a listening position at the middle of the mesh's longest side. Blue lines show sound vertex projections to the listener by linking each sound vertex to the listener position, and small red cubes are placed at the location on each line where nearest surface distance attenuation would have the associated sound component localized in space. The nearest point on the mesh's surface is visualized as a larger, brighter red cube. All red cubes are equidistant from the listener position.

Figure 7.2 shows the same elements however the car mesh is rotated 45 degrees and the point cloud of red cubes is denser, closer, and of smaller extent.

Figure 7.3 shows the same elements except the car mesh is now oriented so that it should be heard as having its smallest possible extent for those constant listener and mesh positions.

Figure 7.4 shows a configuration similar to Figure 7.1 but with a different view. In this view, it can be seen that the surface approximated by the red cubes resembles the profile of the car mesh.

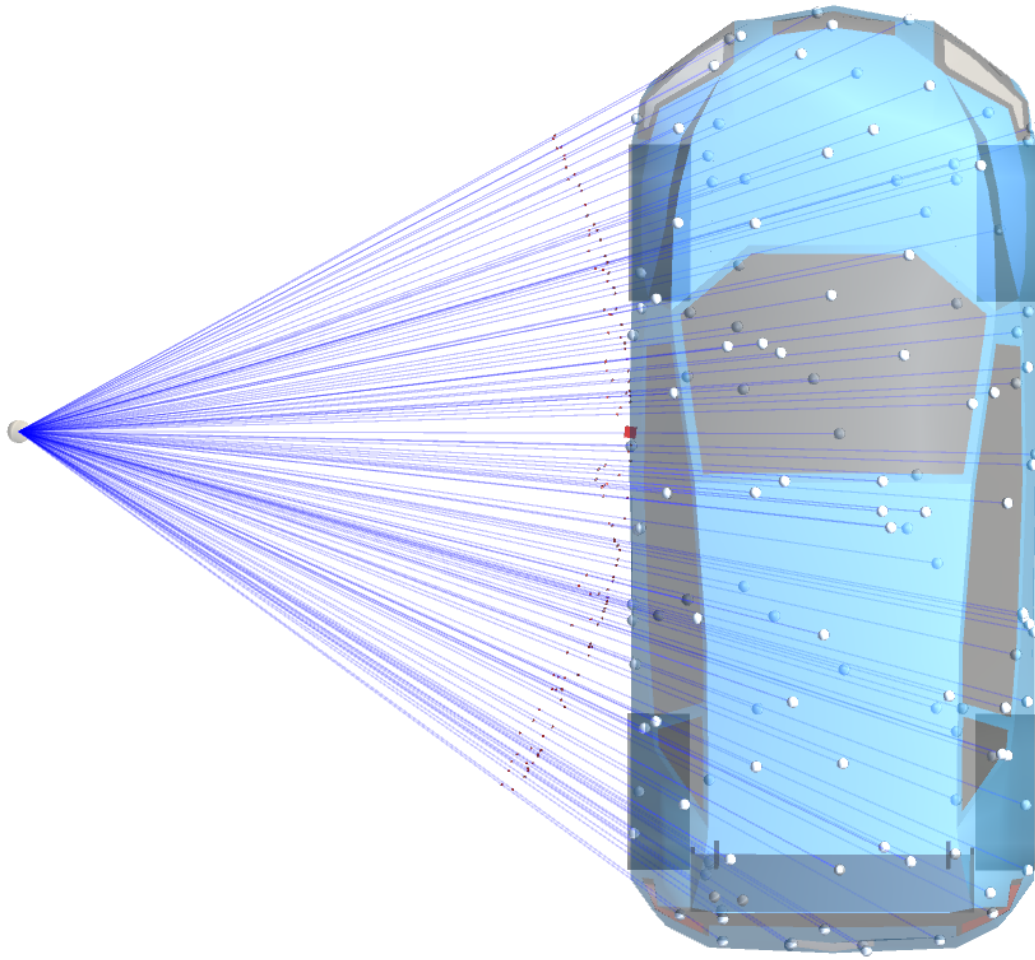


Figure 7.1: A 3D mesh representing a car is heard at its widest extent from the listening position represented by a white sphere on the left. Blue lines point from every sound vertex to the listener position. Small red cubes show the apparent distance that each sound vertex would be localized at. A bright red cube shows the nearest point on the mesh's surface. All red cubes are equidistant to the listener position.

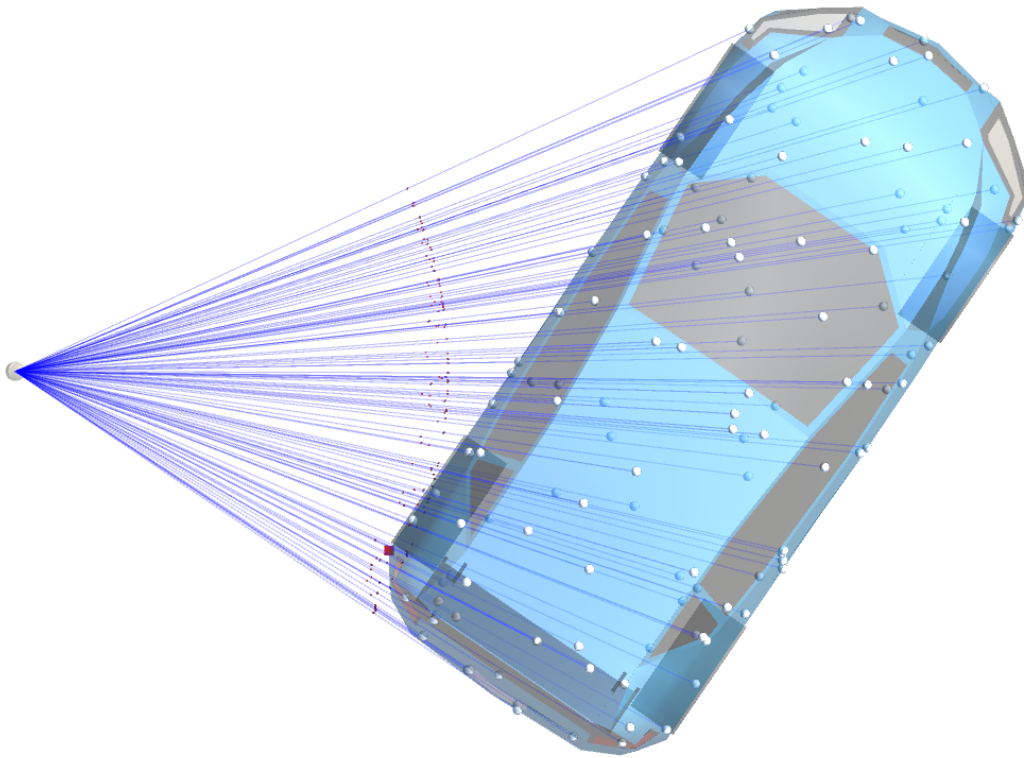


Figure 7.2: A 3D mesh representing a car is heard at an angle from the listening position represented by a white sphere on the left. Blue lines point from every sound vertex to the listener position. Small red cubes show the apparent distance that each sound vertex would be localized at. A bright red cube shows the nearest point on the mesh's surface. All red cubes are equidistant to the listener position.

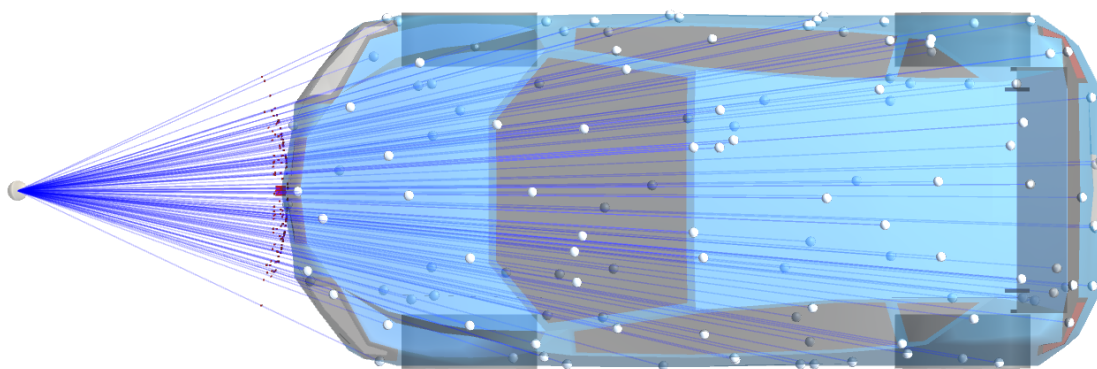


Figure 7.3: A 3D mesh representing a car is heard at its smallest extent from the listening position represented by a white sphere on the left. Blue lines point from every sound vertex to the listener position. Small red cubes show the apparent distance that each sound vertex would be localized at. A bright red cube shows the nearest point on the mesh's surface. All red cubes are equidistant to the listener position.

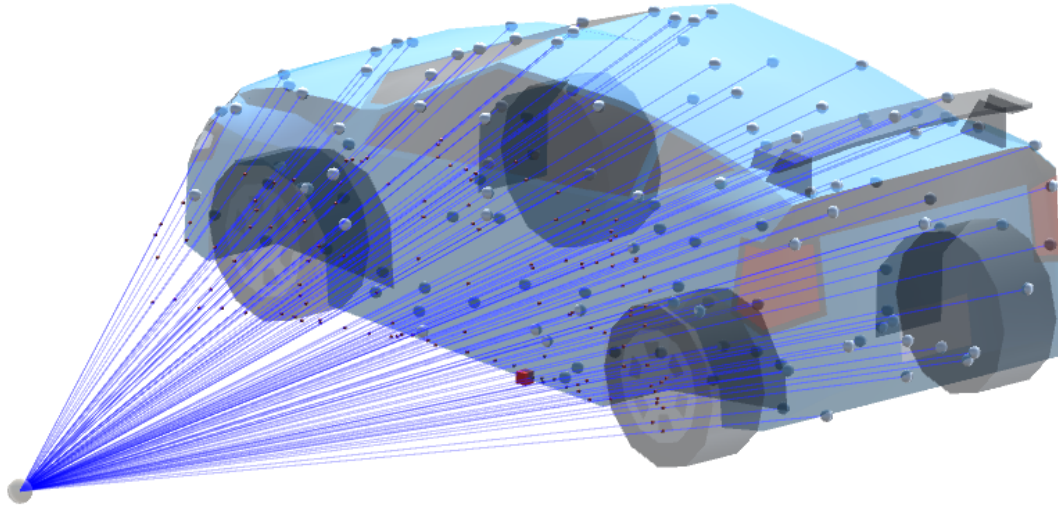


Figure 7.4: A 3D mesh representing a car is heard at its widest extent from the listening position represented by a white sphere on the left. Blue lines point from every sound vertex to the listener position. Small red cubes show the apparent distance that each sound vertex would be localized at. A bright red cube shows the nearest point on the mesh's surface. The perspective shows that the projected image made from the red cubes resembles the profile of the 3D car mesh.

7.1 Hybrid Distance Attenuation Method

Flattening distance attenuation across signal components is an easy way to avoid spectral coloration when using frequency decomposition for pointwise spatial extent synthesis. However, for concave geometry flattening distance attenuation permits a degenerate case where getting close to a sound object's surface produces an incorrect loudness distribution.

In Figure 7.5 a concave geometry with flattened nearest surface attenuation is shown with where its component-point sources are located. The spatial extent distribution shown in that configuration is suitable for a decent spatial extent effect. However, in Figure 7.6 the degenerate case with an incorrect loudness distribution is shown. The component-point sources are all the same loudness, but they appear more densely on the opposite side of the listener from which the nearest surface appears.

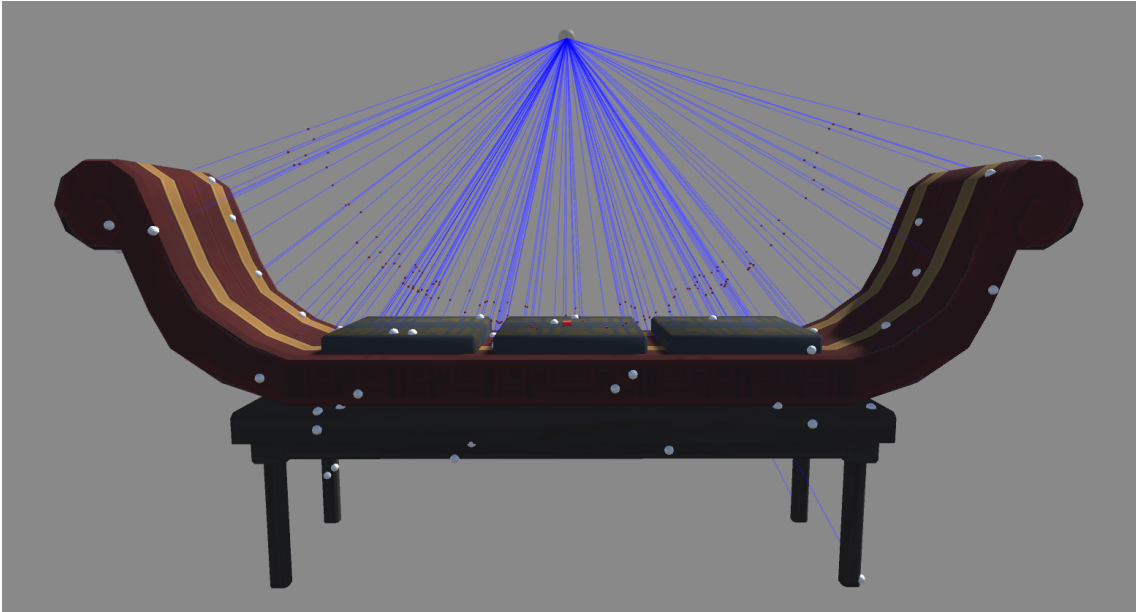


Figure 7.5: A listener position (white sphere touching all blue lines) experiences an acceptable spatial extent for concave geometry. Each small dark red cube represents the effective position that each component-point source is rendered at.

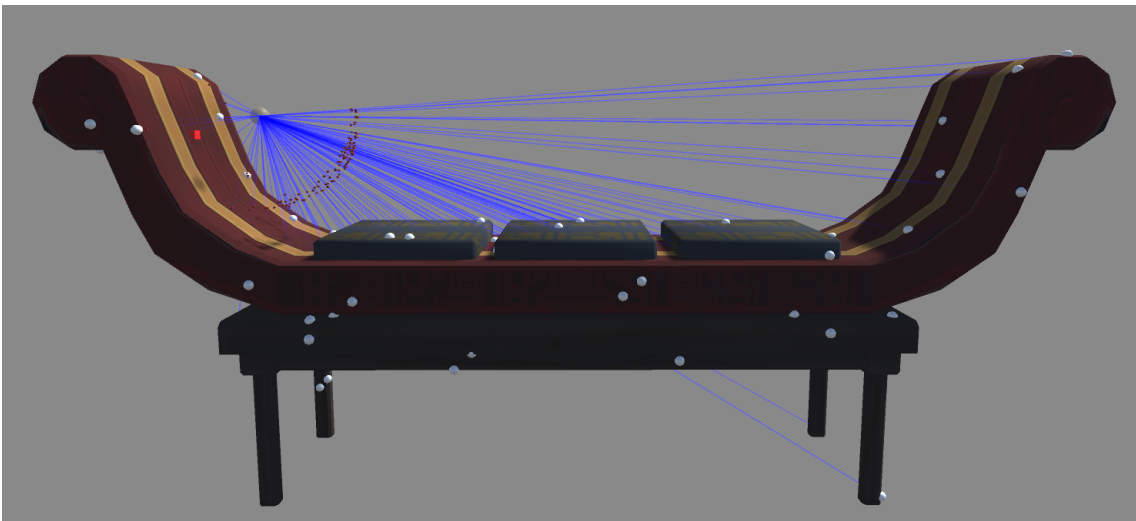


Figure 7.6: A listener position (white sphere touching all blue lines) has a high concentration of component-points appearing in the opposite direction of the nearest surface. The point on the nearest surface is designated by a large bright red cube.

An experimental method for mitigating this poor rendering case was developed that is a hybrid of component-pointwise and nearest surface attenuation models. The main idea is that the nearest surface should be the loudest and most apparent in the spatial field while the spatial extent is still represented in the observer's listening sphere. This idea was attempted by applying component-pointwise distance attenuation as normal, then finding the difference between the nearest surface attenuation and each

component's attenuation. Then, each component is spatialized a second time at the nearest point on the surface, but the gain is set to be that difference between the nearest surface attenuation and the component's former attenuation. In this way, "make-up gain" is applied so that the sum of every component's representation in the sound field is made whole and little spectral coloration will occur.

This hybrid method is visualized in Figure 7.7. The blue lines linked between the listener position and every component-point source are faded according to how much distance attenuation they have applied to them. The more transparent a line is, the more make-up gain it will have applied when being spatialized from the location of the large bright red cube. It can be seen that the loudness distribution becomes much more in favor of the nearest surfaces while still maintaining elements of the full sound object's extent.

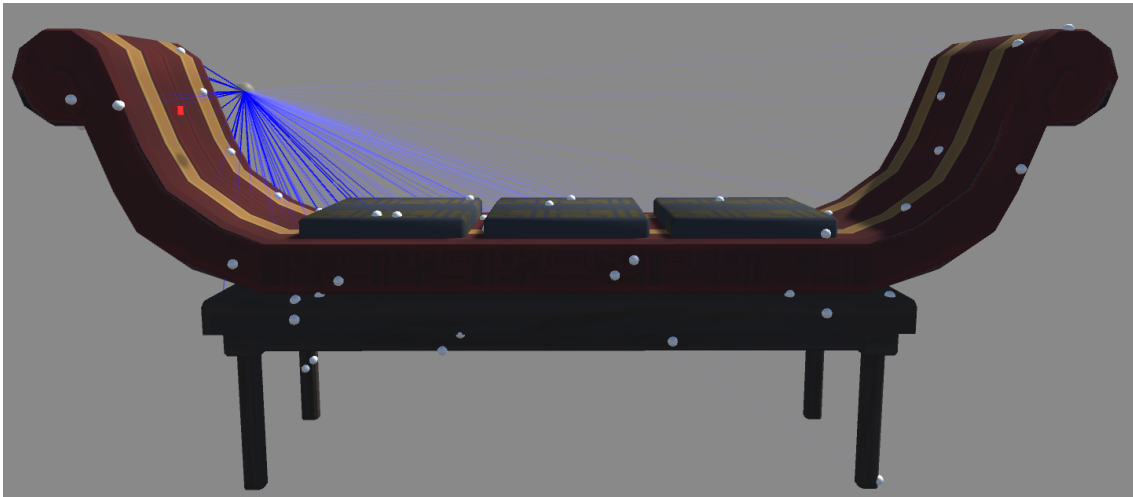


Figure 7.7: *A listener position experiences a loudness distribution that is more suitable for the concave geometry. The blue lines attached to each component-point source are more transparent the quieter the point sources are heard. Make-up gain is emitted from the point on the nearest surface designated by a large bright red cube.*

Now, this hybrid method is not without its own drawbacks. For one, the nearest distance to the geometry's surface will remain continuous when listener or source object are moving, but the point in space on the geometry's surface is not guaranteed to move continuously. This means that when the nearest surface point may jump drastically to another position, then the apparent loudness distribution will change immediately and significantly. Secondly, the hybrid method's apparent loudness distribution cannot be said to be entirely accurate, since the loudness of the redistributed make-up gain points may vary unnaturally for how some nearby extensive sound object should sound in reality.

8 Implementation: Real-Time Virtual-World Spatial Extent Synthesizer

There are many words to describe what this implementation is capable of which could lead to many extended acronyms, but a simple handle will be useful for reading going forward. ‘bsynth’ is the name of the implementation’s project files, named after “B-Format Synthesizer”. I find ‘bsynth’ to be a suitable working name for the implementation; simply read it as “bee-synth”.

‘bsynth’ is a virtual-world spatial extent synthesizer which renders to ambisonics B-format signals and runs in real-time as an audio plug-in for various host applications. Let’s review the listed qualifiers to clarify:

1. Virtual-World

‘bsynth’ models a virtual world by maintaining state for positions, orientations, and geometry of virtual sound sources and a virtual listener avatar. It also provides physical approximations for certain phenomena such as continuous object movements by use of position interpolation, and acoustic propagation delay with Doppler Effect by use of an optional fractional delay line.

2. Spatial Extent Synthesizer

‘bsynth’ is properly a spatial extent synthesizer because it spatially extends mono sound signals rather than manipulating spatial audio formats which may already contain spatial extent.

3. Renders to Ambisonics B-format Signals

‘bsynth’ renders (or synthesizes) ambisonics B-format signals. This led to the workable, but somewhat vague handle of “bsynth”.

4. Runs in Real-Time as an Audio Plug-in

‘bsynth’ is portable and capable of short-time processing in order to avoid being locked into an offline and proprietary implementation that is often seen in academic research.

8.1 Real-Time Plug-In Architecture

A plug-in architecture was chosen for ‘bsynth’ to accomplish real-time audio signal processing with arbitrary audio inputs and to maintain portability. In this architecture, a host application dictates the properties and flow of digital audio while client instances (plug-ins) are subject to those properties and flow. Plug-ins are initialized with the properties of an audio stream (sample rate and block size) and then they are repeatedly asked to handle short duration buffers of audio, called “blocks”, provided by the host. A plug-in may modify a block of audio that it is given, or it may simply read the block for analysis purposes. It is common among plug-in hosts that the same number of input and output channels are provided to its plug-ins in the audio blocks. This is because plug-ins are typically placed in series within a mixer channel

that models traditional mono or stereo effects chains. However, some VST hosts, like Reaper and Max, allow different amounts of input and output channels to be specified.

For this implementation, the number of input channels is dictated by the number of sound sources desired (maximum of 64) and the number of output channels is dictated by the number of orthogonal components within a B-format signal of a chosen spherical harmonic order (maximum of 7th order requiring 64 output channels). At least one input channel is required for any audible processing to occur. Four output channels are necessary for the default setting of first-order ambisonics B-format output.

8.2 VST and JUCE

VST (Virtual Studio Technology) is an audio plug-in software interface and SDK created by software and hardware company Steinberg. It defines an architecture for audio handling, MIDI handling, graphical user interfaces, and parameter passing for audio processing plug-ins. The VST standard is well supported by applications running on Windows, Mac, and Linux operating systems.

JUCE is a cross-platform application framework and SDK. It is well known for its abstraction of and adherence to multiple audio plug-in interface specifications. One of the plug-in standards it implements is VST. The JUCE SDK and its IDE, Projucer, were used to simplify the creation of ‘bsynth’ as a VST plug-in.

8.3 Hosts

Host applications, or hosts, maintain the lifetime of the ‘bsynth’ plugin and use it for processing audio. Below are descriptions for two VST hosts, and the game engine Unity which is also used as a host.

8.3.1 Reaper

Reaper is a fully-fledged Digital Audio Workstation (DAW) capable of hosting VST plug-ins. Reaper also supports up to 64 inputs and 64 outputs in a mixer channel, so it made a great testbed during the initial development of ‘bsynth’.

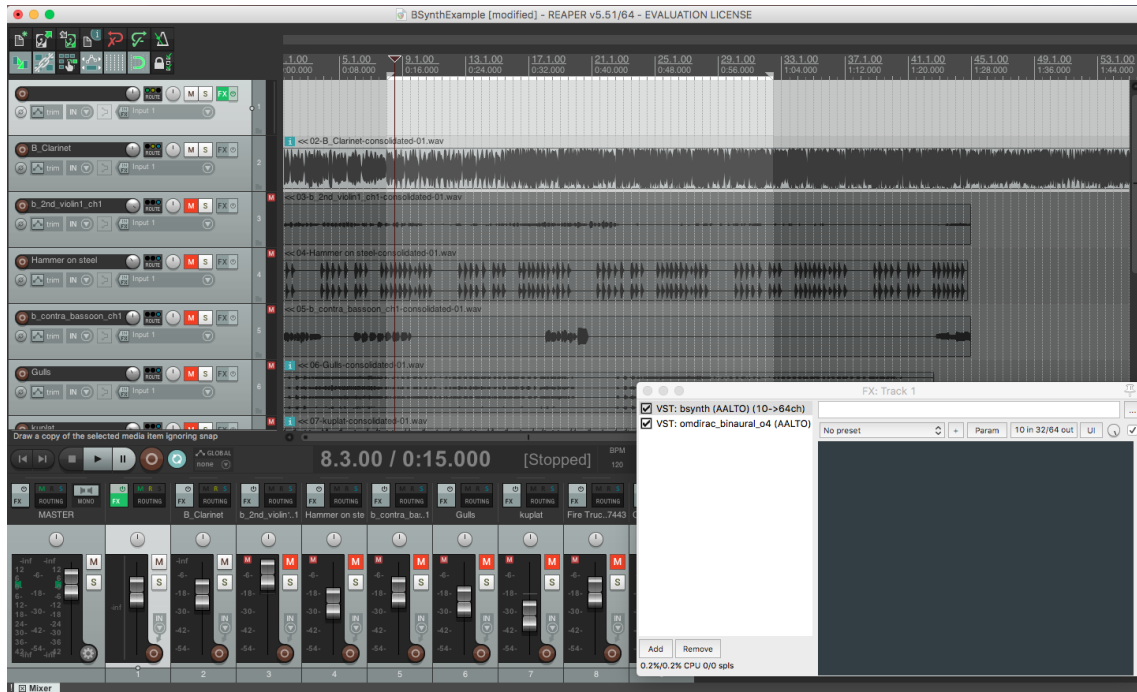


Figure 8.1: A screenshot of Reaper showing multiple sound sources being routed into ‘bsynth’ which outputs B-format signals then into ‘omdirac_binaural_o4’ in order to produce binaural audio from spatially extended sound sources.

8.3.2 Max

Max is a graphical programming language and runtime environment for multimedia purposes. Max, often appearing in audio signal processing research (Schacher and Kocher, 2006), has its own specifications for plug-ins called ‘externals’. With externals, a tight integration can be made between Max and custom DSP or UI features. However, Max can also host VST plug-ins through an object titled ‘vst~’. ‘bsynth’ implements an inter-process command interface that replaces the role of Max messages and VST parameters, so it remains nearly as flexible as a native Max external without message support.

‘bsynth’ can be set up by creating a ‘vst~’ block in a Max patch and then specifying the number of input channels, the number of output channels, and the ‘bsynth’ VST filename.

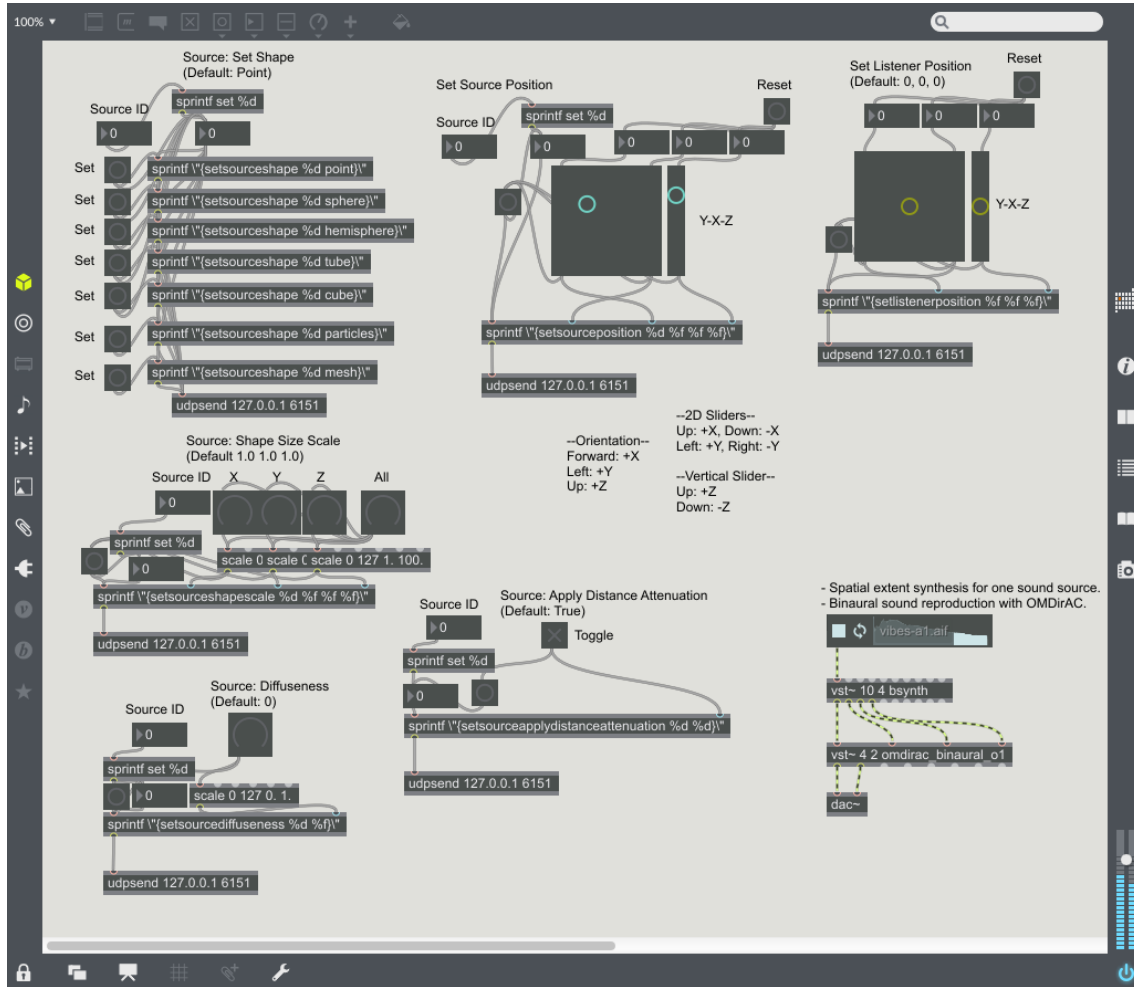


Figure 8.2: Max shown with many user interface elements that create commands for ‘bsynth’ which are to be sent with ‘udpsend’ blocks. In the lower-right area, ‘vst~’ blocks are shown hosting ‘bsynth’ and ‘omdirac_binaural_o1’ in order to produce binaural audio from spatially extended sound sources.

8.3.3 Unity (Game Engine)

Unity is a cross-platform game engine, which means that it provides graphics rendering, audio processing, physics simulating, collision detection, and game entity management capabilities on multiple computing platforms. Unity compiles executables on each platform it supports and packages game assets (textures, 3D models, game sounds, and scripts) for deployment.

By design, Unity is not capable of hosting VST plug-ins. This is because of portability and distribution license concerns for VST plug-ins being bundled with games. Instead, Unity provides an implementable C-header specification for shared libraries to be loaded and used as audio plug-ins. ‘bsynth’ is retrofitted to be both a VST and a Unity audio plug-in in the same shared library binary.

For Unity projects, the ‘bsynth’ binary must be placed into a game project’s plugins directory (`ProjectName/Assets/Plugins/CPUArchitecture/`) before Unity will recognize its existence. In order to use ‘bsynth’ in a project, one audio mixer group must have the ‘bsynthAalto’ effect in its effects chain, and one or more separate groups must have the ‘bsynthAaltoSource’ effect in their effects chains. During each audio processing block, ‘bsynthAaltoSource’ effects read the input audio from their groups so they can route the audio through shared memory to the ‘bsynthAalto’ effect. The ‘bsynthAalto’ effect performs all spatialization and renders to binaural head-related-transformed audio by use of the OMDirAC binaural renderer that was mentioned in Chapter 4. Audio sources in Unity should be used regularly, but the output mixer group needs to be set to a mixer group that contains a ‘bsynthAaltoSource’ effect configured for the source ID that the sound source is destined for.

Figures 8.3, 8.4, and 8.5 show some of the views that are seen when using ‘bsynth’ in Unity.

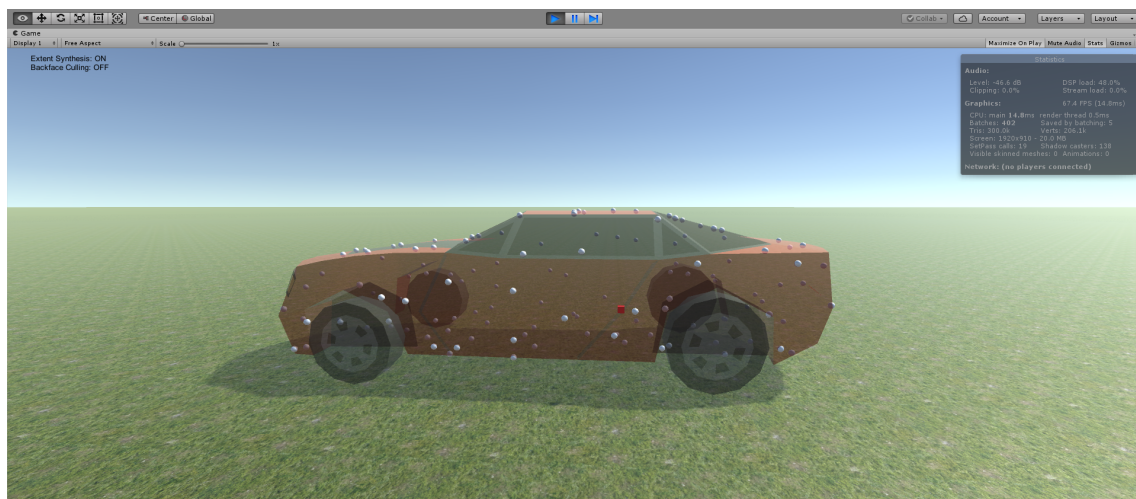


Figure 8.3: Unity’s editor shown in ‘Play’-mode. A car’s 3D model is shown with component-point sources distributed across its surface as small white spheres, and a red cube shows the nearest point to the 3D model’s surface from the listener position.

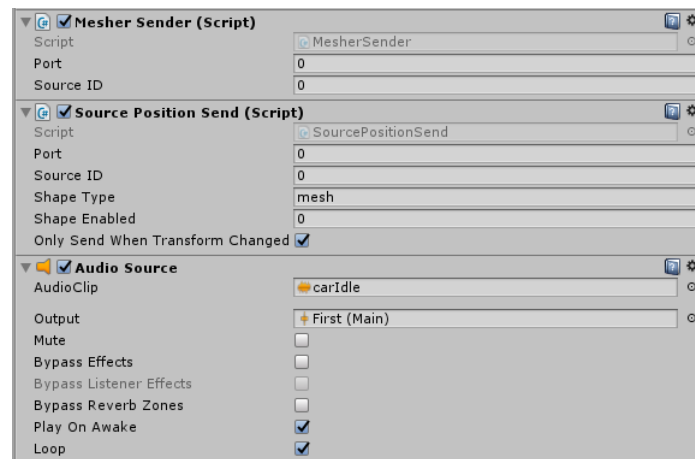


Figure 8.4: A section of Unity's 'Inspector' pane for a sound source object. 'Mesher Sender' is a script that uploads the 3D mesh from Unity to 'bsynth'. 'Source Position Send' is a script that keeps the position and orientation of a sound source updated in 'bsynth', and it also specifies the shape type and when to update the sound source orientation.

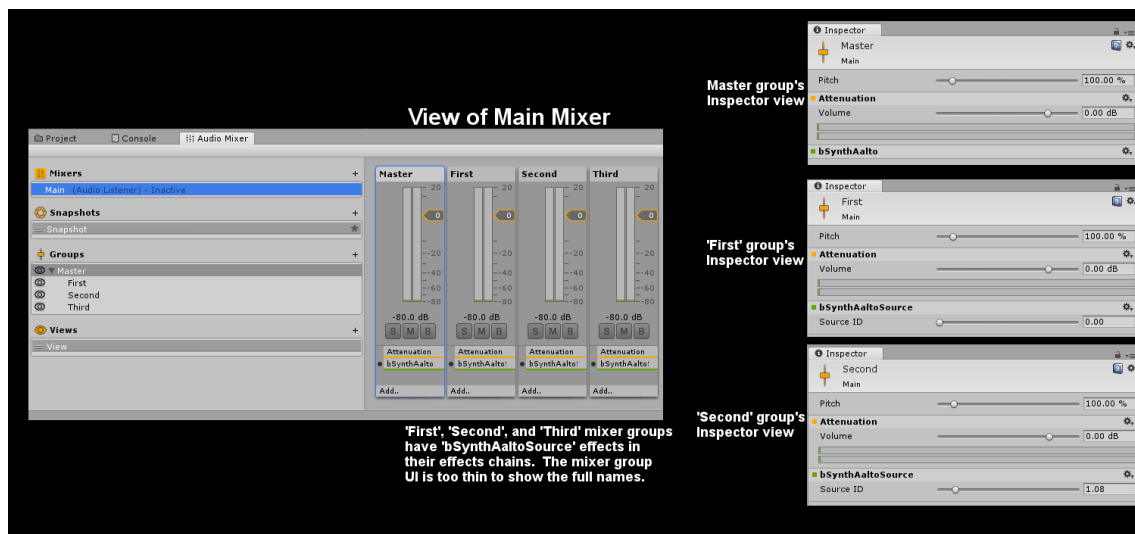


Figure 8.5: A view of Unity's audio mixer and the 'Inspector' views for certain mixer groups. The 'Source ID' for 'bSynthAaltoSource' effects is a floating-point value that is rounded down to become the source ID which is handled by that effect (a result of limitations from Unity audio plug-in parameters at the time of making).

8.4 Command-Interface and Virtual State

VST plug-ins often have customized graphical user interfaces and named parameters for controlling the behavior of the plug-in. Similarly, Max externals live in the Max user interface/runtime environment and can accept Max messages; accomplishing the same purpose. For 'bsynth' however, a fully featured user interface with support for controlling the 3D placement and orientation of sound sources along with the

modification of their parameters would take a lot to develop. And if controlling the plug-in relied only on a user interface for configuration, then the plug-in's flexibility would be limited to only manually designed sound environments. Instead of any user interface, a command-based protocol approach was developed that is similar to Max messages so that control of the synthesis engine could be scripted and manipulated by a host application or a separate application running on the local PC.

A single local-loopback UDP server socket is maintained during the lifetime of the 'bsynth' plug-in which can receive plaintext commands from any locally running application. This type of OS-wide protocol is called 'interprocess communication'. During development, 'bsynth' was most often hosted by Reaper while commands were sent from a Max test patch using the 'udpsend' object. Later in development, Unity became able to host 'bsynth' and use C# scripted sockets to stream Unity's game state into the plug-in's world state.

Every exposed feature of 'bsynth' has a command consisting of a command name and then either a fixed or variable number of input arguments, all of which must be a suitable string representation for their datatypes. Commands are surrounded by curly brace characters to signify command beginnings and ends. An example command is `{SetSourceVertices 3 0.1 0.2 -2.0}`, which would set the sound source with ID "3" to have one vertex, $\mathbf{v}_{\text{vertex}} = (0.1, 0.2, -2.0)$.

The state of the virtual world consists of global parameters, multiple source states, and a listener orientation. A single source state contains a sound source's position, shape, scale, rotation, diffuseness, and toggles for various treatments that the source should receive. There is a source state structure for every input audio channel in an instance of 'bsynth'.

8.5 Global State and Parameters

The following is a list of many of the relevant global state variables and modifiable parameters within 'bsynth'. Global parameters can be changed to affect all source rendering.

Listener Position

A 3D position representing where the center of the listener's head is. In other words, the center position of a virtual B-format microphone.

Facing

A 3D vector which specifies the forward direction of a virtual B-format microphone placed at the listener position. This can be changed to allow the sound designer's preference for a forward/backward axis. This is not required to change with the listener's orientation, because directionality can be manipulated in the decoding stage. When unchanged, the synthesized B-format signals will remain axis-aligned. That should be no issue because HRTF-based B-format decoders can face a virtual body in any direction during decoding. And also B-format signals can be rotated

before they are decoded for loudspeaker reproduction if needed.

Sample Rate

The VST host or Unity specify a sample rate that the B-synth plugin accepts.

Samples per Block

The VST host or Unity specify how many samples are to be processed with every ‘process’ callback. Knowing this before processing allows the synthesizer to allocate sufficient working memory for processing.

Spherical Harmonic Order

This is the global setting for the order of B-format synthesis that is to be rendered. The spherical harmonic order determines the number of output channels in the B-format signal. By default, this is set to 1 (first order) which produces 4 output channels (W, X, Y, and Z). Anywhere from orders 0 through 7 are supported by the engine. Order 0 produces 1 output channel (just W). Order 7 produces 64 output channels (an omnidirectional signal and 63 more specific directional signals).

Enable Reverb and Reverb Order

This setting enables a spatial FIR convolution reverb. The reverb also has an order parameter that determines how many spatial components the reverb will be performed on (similar in number to the B-format synthesis spherical harmonic order). For every channel that will be rendered, an FIR filter can be specified. In this way, reverb sound can be decoded with directionality.

Normal Distance

The distance calculations in the engine are designed to be unit agnostic meaning that there is no preference for meters, centimeters, or feet as the true measure behind position and speed quantities. Because of this, a global ‘normal distance’ can be specified. This distance represents the standard recording distance for the sound sources. Within this distance to the listener, the signal level from sound sources will not get louder in order to prevent over amplification of the source signals. Beyond this normal distance, the inverse-square law for distance attenuation is applied with this normal distance as the reference distance with which to calculate the reduced gain.

8.6 Virtual Sound Source State and Parameters

8.6.1 State

Target Position and Interpolation Rate

‘Target Position’ was implemented so that continuous object movement through space would be simulated. When a source is commanded to move (by relative or absolute position) its position is interpolated between its current position and a target position. The rate at which the position changes is called the ‘Interpolation

Rate’.

This behavior serves to prevent discontinuities in the audio due to teleportation-like position updates. This behavior also aids with rendering the Doppler Effect, which is synthesized using fractional delays in a ring buffer for each sound source.

If an object is commanded to change position before its current position has reached the target position, then the target position is updated and interpolation continues in the direction of the new target.

Two-pass processing with crossfaded signals is a common method for masking the discontinuities caused by parameter changes in audio processing algorithms (Wenzel et al., 2000; Gnegy, 2014). However for position changes, this kind of crossfading was avoided in order to save resources in the case where many sources may be moving simultaneously.

Current Position

This is the central position for a sound source. This position is used for point sources as well as spatially extended sources. With spatial extension, a cloud of offsets from the current position approximate the sound source’s surface.

The vector pointing from the listener position to some 3D position which is offset from the current position is used to determine relative polar coordinates (r, θ, ϕ) . These polar coordinates are used as input into an ambisonic panner for a single point source and for spatially positioned frequency components of spatially extended sources.

Source position can be controlled by four different commands: **SetSourcePosition**, **SetSourcePositionInstant**, **MoveSource**, and **MoveSourceInstant**. The “instant” variants of these commands apply the position changes without any interpolation of the position-state values over time by setting both the current position and target position to the same value. The “set” variants set the target position directly, and the “move” variants apply a delta to the current target position.

Ring Buffer

When sound propagation simulation is enabled, each sound source maintains a ring buffer for use as a delay line. Fractional delays are created from linear sample interpolation and then emplaced in the discrete time-step samples of the ring buffer. Delay is calculated according to the distance from the listener to a sound source and with a hardcoded speed of sound (which should have become a configurable global parameter). This creates a Doppler Effect as well as time delay due to propagation. The maximum delay length is hardcoded to 3 seconds.

8.6.2 Parameters

Shape

A sound source can have one ‘shape’ specified which will determine how its spatial

extent is distributed. The command to set a source shape is: `SetSourceShape`. The ‘shape’ types are follows:

- **Point** (Default) – A point source shape. No spatial extent is formed, but spatialization occurs from the source position by using an ambisonic panner.
- **Mesh** – Creates spatial extent by spreading point sources onto an arbitrary 3D mesh. Before this shape will perform extent synthesis, a mesh needs to be provided by two commands: `SetSourceVertices` and `SetSourceVertexIndices`. Point distribution follows the mesh-surface random sampling method that was detailed in Chapter 6. The vertices and vertex indices should follow industry standard practice for tri-meshes, where “vertices” is the set of all points and “vertex indices” is the set of tri-faces in order where every three indices refer to the three vertices by-index which form a single tri-face.
- **Particles** – Creates spatial extent by using the source’s vertices as point sources, and randomly distributing the source’s decomposition components amongst them. Before this shape will perform extent synthesis, a set of vertices needs to be provided by the command `SetSourceVertices`.
- **Sphere** – Creates spatial extent by analytical spreading of point sources onto a sphere surface.
- **Hemisphere** – Creates spatial extent by analytical spreading of point sources onto a hemisphere surface.
- **Cube** – Creates spatial extent by analytical spreading of point sources onto a cube surface.
- **Tube** – Creates spatial extent by analytical spreading of point sources onto a tube (cylinder with no ends) surface.

Shape Scale

A source’s shape geometry can be scaled along its three axes x , y , and z . Scaling happens before rotation when rendering. Shape scale is set by the command: `SetSourceShapeScale`.

Shape Rotation

A source’s shape geometry can be rotated by specifying an axis to rotate around as a 3D unit vector, and then an angle representing the magnitude of rotation around the specified axis. This representation of rotation is called the ‘axis-angle representation’, and it can represent any rotation or sequence of rotations of a rigid body. Unity and other game engines provide this rotation representation readily for game entities in their scripting languages. Shape rotation is set by the command: `SetSourceShapeRotation`.

Distance Attenuation Method

A source may have five different types of distance attenuation calculated. The source's distance attenuation method is set by the command `SetSourceDistanceAttenuationMethod`. More details on the methods are available in Chapter 7. The methods are:

- **Pointwise** – Distance attenuation can be different for every component of the spatially extended sound. Distance attenuation is calculated individually for each component-point source that is spatially distributed.
- **Origin** – Distance attenuation is calculated from the distance between the sound source origin and the listener position. This attenuation is applied to all component-point sources equally.
- **NearestSoundVertex** – Distance attenuation is calculated from the distance to the nearest component-point source (sound vertex) in relation to the listener position. This attenuation is applied to all component-point sources equally.
- **NearestSurface** – Distance attenuation is calculated from the distance to the nearest point on the source object surface. This calculation is quick and trivial for analytical geometries (sphere, hemisphere, cube, and tube). However, this calculation is very CPU intensive for Mesh shaped sources, since the nearest point on every face must be calculated. However, with occlusion culling the nearest vertex could be used to find a small set of prospective faces.
- **PointwiseSurfaceHybrid** – Distance attenuation is calculated individually for each component-point source that is spatially distributed, and then the distance to the nearest surface is found to calculate another attenuation factor. Each component-point source will have a gain factor that is less than the nearest surface gain factor. To restore the overall frequency magnitude spectrum, each component is spatialized at the nearest point on source shape with a gain factor that is the difference between the nearest surface gain and the component's original position gain.

Apply Distance Attenuation

For a skybox effect or debug purposes, a sound source can have distance attenuation deactivated. In the case of a skybox, a shape sized to be larger than the virtual walking grounds or a source that follows the listener position could be used to provide position-independent ambience. Setting whether distance attenuation is applied or not is done by the command: `SetSourceApplyDistanceAttenuation`.

Diffuseness

Diffuseness is a parameter with values in the interval $[0, 1]$ which balances rendered sound between directional (non-diffuse) and non-directional (diffuse) quantities in the B-format output signal. Diffuseness is applied to all source decomposition components

equally, although in nature it is much more independent for each component. Having this diffuseness parameter greater than zero can help mask degenerate cases during synthesis where only a few component-point sources are off to one side, or when near-frequency components are spread widely apart (Laitinen et al., 2012).

9 Conclusion

A virtual-world spatial audio renderer was implemented that features spatial extent synthesis for virtual sound source objects. Listeners who experience decodings of this implementation's output B-format signals should be able to sense the apparent sizes of virtual sound source objects that are present within a virtual environment. The implementation is able to interface with a few 3D graphics renderers, namely the Unity game engine and Jitter extensions for Max. Extra features such as spatial extent synthesis for arbitrary 3D mesh sound source geometry and new methods for distance attenuation of spatially extensive sound sources were included.

Quasi-random Halton sequences were used to sample geometry surfaces with low-discrepancy in order to minimize audible spatial artifacts in the results of spatial extent synthesis. The implementation saw experiments with rendering opaque sound objects by use of back-face culling, while discussing a better form using occlusion culling. Concerns with loudness density in spatial distribution with application of certain distance attenuation models were discussed as well as an attempted method for mitigating those concerns. Also, the implementation's details such as plug-in architecture, host applications, command interface, behavior, state, and parameters were all discussed.

Improvements to the spatial extent synthesis and distance attenuation methods can be made:

A form of geometry-wise distribution which doesn't require points, but rather deals with apparent surfaces entirely may be more desirable if it can be accomplished at low computational cost and with few audible artifacts.

As for distance attenuation, there should be more perceptually motivated methods that are in concordance with experiences of real world sound objects of significant size. For one example, line and plane sound sources don't behave in the same way as point sources in terms of distance attenuation until they are significantly far away from a listener. This distance attenuation behavior should be automatically rendered for arbitrary geometry in the same way that spatial extent synthesis was shown to be possible.

Finally, the virtual-world audio processing could have included spatial reverb, general occlusion, and non-uniform source radiation patterns as features to increase listener immersion.

Bibliography

- Ahveninen, J., Kopčo, N. and Jääskeläinen, I. P. (2014). Psychophysics and neuronal bases of sound localization in humans, *Hearing Research* **307**: 86–97.
- Algazi, V. R., Duda, R. O., Thompson, D. M. and Avendano, C. (2001). The cipic hrtf database, *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*, IEEE, pp. 99–102.
- Blauert, J. (1997). *Spatial hearing: the psychophysics of human sound localization*, MIT press.
- Boashash, B. and Black, P. (1987). An efficient real-time implementation of the wigner-ville distribution, *IEEE transactions on acoustics, speech, and signal processing* **35**(11): 1611–1618.
- Callahan, M. (1977). Acoustic signal processing based on the short-time spectrum, *The Journal of the Acoustical Society of America* **61**(S1): S51–S52.
- Chan, D. (1982). A non-aliased discrete-time wigner distribution for time-frequency signal analysis, *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 7, IEEE, pp. 1333–1336.
- Daniel, J. (2000). *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*, PhD thesis, University of Paris.
- de Vries, D., Hulsebos, E. M. and Baan, J. (2001). Spatial fluctuations in measures for spaciousness, *The journal of the Acoustical Society of America* **110**(2): 947–954.
- Delprat, N., Escudié, B., Guillemain, P., Kronland-Martinet, R., Tchamitchian, P. and Torresani, B. (1992). Asymptotic wavelet and gabor analysis: Extraction of instantaneous frequencies, *IEEE transactions on Information Theory* **38**(2): 644–664.
- Faller, C. and Merimaa, J. (2004). Source localization in complex listening situations: Selection of binaural cues based on interaural coherence, *The Journal of the Acoustical Society of America* **116**(5): 3075–3089.
- Farag, H., Blauert, J. and Abdel Alim, O. (2003). Psychoacoustic investigations on sound-source occlusion, *Audio Engineering Society Convention 114*, Audio Engineering Society.
- Gardner, B., Martin, K. et al. (1994). Hrtf measurements of a kemar dummy-head microphone.
- Gerzon, M. A. (1973). Periphony: With-height sound reproduction, *Journal of the Audio Engineering Society* **21**(1): 2–10.

- Gerzon, M. A. (1980). Practical periphony: The reproduction of full-sphere sound, *Audio Engineering Society Convention 65*, Audio Engineering Society.
- Gnegy, C. (2014). Collidefx: A physics-based audio effects processor., *NIME*, pp. 427–430.
- Halton, J. H. (1964). Algorithm 247: Radical-inverse quasi-random point sequence, *Communications of the ACM* **7**(12): 701–702.
- Heller, A., Lee, R. and Benjamin, E. (2008). Is my decoder ambisonic?, *Audio Engineering Society Convention 125*, Audio Engineering Society.
- Holt, R. E. and Thurlow, W. R. (1969). Subject orientation and judgment of distance of a sound source, *The Journal of the Acoustical Society of America* **46**(6B): 1584–1585.
- James, D. L., Barbič, J. and Pai, D. K. (2006). Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources, *ACM Transactions on Graphics (TOG)*, Vol. 25, ACM, pp. 987–995.
- Käsbach, J. (2016). *Characterizing apparent source width perception*, PhD thesis, Technical University of Denmark, Lyngby.
- Laitinen, M.-V., Pihlajamäki, T., Erkut, C. and Pulkki, V. (2012). Parametric time-frequency representation of spatial sound in virtual worlds, *ACM Transactions on Applied Perception (TAP)* **9**(2): 8.
- Noisternig, M., Musil, T., Sontacchi, A. and Holdrich, R. (2003). 3d binaural sound reproduction using a virtual ambisonic approach, *IEEE International Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2003. VECIMS'03. 2003*, IEEE, pp. 174–178.
- Oppenheim, A. V. (1970). Speech spectrograms using the fast fourier transform, *IEEE spectrum* **7**(8): 57–62.
- Osada, R., Funkhouser, T., Chazelle, B. and Dobkin, D. (2002). Shape distributions, *ACM Transactions on Graphics (TOG)* **21**(4): 807–832.
- Pihlajamäki, T., Laitinen, M.-V. and Pulkki, V. (2013). Modular architecture for virtual-world parametric spatial audio synthesis, *Audio Engineering Society Conference: 49th International Conference: Audio for Games*, Audio Engineering Society.
- Pihlajamäki, T. and Pulkki, V. (2012). Projecting simulated or recorded spatial sound onto 3d-surfaces, *Audio Engineering Society Conference: 45th International Conference: Applications of Time-Frequency Processing in Audio*, Audio Engineering Society.

- Pihlajamäki, T., Santala, O. and Pulkki, V. (2014). Synthesis of spatially extended virtual source with time-frequency decomposition of mono signals, *Journal of the Audio Engineering Society* **62**(7/8): 467–484.
- Politis, A., McCormack, L. and Pulkki, V. (2017). Enhancement of ambisonic binaural reproduction using directional audio coding with optimal adaptive mixing, *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, pp. 379–383.
- Potard, G. and Burnett, I. (2004). Decorrelation techniques for the rendering of apparent sound source width in 3d audio displays, *Proc. Int. Conf. on Digital Audio Effects (DAFx'04)*.
- Pulkki, V. (2007). Spatial sound reproduction with directional audio coding, *Journal of the Audio Engineering Society* **55**(6): 503–516.
- Pulkki, V., Delikaris-Manias, S. and Politis, A. (2018). *Parametric time-frequency domain spatial audio*, Wiley Online Library.
- Rayleigh, L. (1907). Xii. on our perception of sound direction, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **13**(74): 214–232.
- Schacher, J. C. and Kocher, P. (2006). Ambisonics spatialization tools for max/msp, *Omni* **500**(1).
- Schmele, T. and Sayin, U. (2018). Controlling the apparent sourcesize in ambisonics using decorrelation filters, *Audio Engineering Society Conference: 2018 AES International Conference on Spatial Reproduction-Aesthetics and Science*, Audio Engineering Society.
- Vilkamo, J. (2015). Alias-free short-time fourier transform - a robust time-frequency transform for audio processing, <https://github.com/jvilkamo/afSTFT>. Accessed: 30-Mar-2019.
- Wang, D. and Brown, G. J. (2006). *Computational auditory scene analysis: Principles, algorithms, and applications*, Wiley-IEEE press.
- Wenzel, E. M., Miller, J. D. and Abel, J. S. (2000). A software-based system for interactive spatil sound synthesis, Georgia Institute of Technology.
- Yost, W. A. and Gourevitch, G. (1987). *Directional hearing*, Springer.
- Zahorik, P. (2002). Assessing auditory distance perception using virtual acoustics, *The Journal of the Acoustical Society of America* **111**(4): 1832–1846.
- Zahorik, P., Brungart, D. S. and Bronkhorst, A. W. (2005). Auditory distance perception in humans: A summary of past and present research, *ACTA Acustica united with Acustica* **91**(3): 409–420.

A Formulas

A.1 Rodrigues' rotation formula

Rotate a vector \mathbf{v} around an axis described by unit vector \mathbf{k} by an angle θ to find the result \mathbf{v}_{rot} with the formula

$$\mathbf{v}_{\text{rot}} = \mathbf{v} \cos \theta + (\mathbf{k} \times \mathbf{v}) \sin \theta + \mathbf{k}(\mathbf{k} \cdot \mathbf{v})(1 - \cos \theta). \quad (\text{A1})$$

A.2 Heron's formula

To find the area, A , of a triangle with sides of lengths a , b , and c , the basic Heron's formula is

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \quad (\text{A2})$$

where s is the semi-perimeter of the triangle calculated as

$$s = \frac{a+b+c}{2}. \quad (\text{A3})$$